

SAMS
**Teach
Yourself**

- 全球销量逾百万册的系列图书
- 连续十余年打造的经典品牌
- 直观、循序渐进的学习教程
- 掌握关键知识的最佳起点
- “Read Less, Do More”（精读多练）的教学方法
- 以示例引导读者完成最常见的任务

每章内容针对初学者精心设计，**1**小时轻松阅读学习，
24小时彻底掌握关键知识

每章**案例与练习题**助你轻松完成常见任务，
通过**实践**提高应用技能，巩固所学知识

Java

入门经典（第7版）

[美] Rogers Cadenhead
郝记生

目 录

[版权信息](#)

[版权声明](#)

[内容提要](#)

[关于作者](#)

[献辞](#)

[致谢](#)

[前言](#)

[第1章 成为程序员](#)

[1.1 选择编程语言](#)

[1.2 告诉计算机做什么](#)

[1.3 程序的工作原理](#)

[1.4 为什么程序不能正常工作](#)

[1.5 选择一个Java编程工具](#)

[1.6 安装Java开发工具](#)

[1.7 总结](#)

[1.8 问与答](#)

[1.9 测验](#)

[1.10 练习](#)

[第2章 编写第一个程序](#)

[2.1 编写程序所需的工具](#)

[2.2 创建Saluton程序](#)

[2.3 开始输入程序](#)

[2.4 在变量中存储信息](#)

[2.5 保存编写好的程序](#)

[2.6 将程序编译为class文件](#)

- [2.7 修复错误](#)
- [2.8 运行Java程序](#)
- [2.9 总结](#)
- [2.10 问与答](#)
- [2.11 测验](#)
- [2.12 练习](#)

[第3章 Java之旅](#)

- [3.1 第一站: Oracle](#)
- [3.2 去Java学校](#)
- [3.3 在JavaWorld用午餐](#)
- [3.4 在NASA仰望天空](#)
- [3.5 回归正题](#)
- [3.6 到SourceForge去问路](#)
- [3.7 在手机上运行Java](#)
- [3.8 总结](#)
- [3.9 问与答](#)
- [3.10 测验](#)
- [3.11 练习](#)

[第4章 理解Java程序是如何工作的](#)

- [4.1 创建应用程序](#)
- [4.2 向应用程序传递参数](#)
- [4.3 Java类库](#)
- [4.4 总结](#)
- [4.5 问与答](#)
- [4.6 测验](#)
- [4.7 练习](#)

[第5章 在程序中存储和修改信息](#)

- [5.1 语句和表达式](#)
- [5.2 指定变量类型](#)
- [5.3 给变量命名](#)
- [5.4 在变量中存储信息](#)
- [5.5 运算符](#)
- [5.6 使用表达式](#)
- [5.7 总结](#)

[5.8 问与答](#)

[5.9 测验](#)

[5.10 练习](#)

[第6章 使用字符串来交流](#)

[6.1 在字符串中存储文本](#)

[6.2 在程序中显示字符串](#)

[6.3 在字符串中使用特殊字符](#)

[6.4 拼接字符串](#)

[6.5 将其他变量用于字符串中](#)

[6.6 字符串的高级处理](#)

[6.7 导演及演员名单](#)

[6.8 总结](#)

[6.9 问与答](#)

[6.10 测验](#)

[6.11 练习](#)

[第7章](#)

[7.1 if语句](#)

[7.2 if-else语句](#)

[7.3 switch语句](#)

[7.4 三元运算符](#)

[7.5 观察时钟](#)

[7.6 总结](#)

[7.7 问与答](#)

[7.8 测验](#)

[7.9 练习](#)

[第8章 使用循环重复执行操作](#)

[8.1 for循环](#)

[8.2 while循环](#)

[8.3 do-while循环](#)

[8.4 退出循环](#)

[8.5 给循环命名](#)

[8.6 测试计算机的运行速度](#)

[8.7 总结](#)

[8.8 问与答](#)

[8.9 测验](#)

[8.10 练习](#)

[第9章 使用数组存储信息](#)

[9.1 创建数组](#)

[9.2 使用数组](#)

[9.3 多维数组](#)

[9.4 对数组进行排序](#)

[9.5 对字符串中的字符计数](#)

[9.6 总结](#)

[9.7 问与答](#)

[9.8 测验](#)

[9.9 练习](#)

[第10章 创建第一个对象](#)

[10.1 面向对象编程的工作原理](#)

[10.2 对象示例](#)

[10.3 什么是对象](#)

[10.4 理解继承](#)

[10.5 建立继承层次](#)

[10.6 转换对象和简单变量](#)

[10.7 创建对象](#)

[10.8 总结](#)

[10.9 问与答](#)

[10.10 测验](#)

[10.11 练习](#)

[第11章 描述对象](#)

[11.1 创建变量](#)

[11.2 创建类变量](#)

[11.3 用方法来创建行为](#)

[11.4 将一个类放在另一个类中](#)

[11.5 使用关键字this](#)

[11.6 使用类方法和类变量](#)

[11.7 总结](#)

[11.8 问与答](#)

[11.9 测验](#)

11.10 练习

第12章 充分利用现有对象

12.1 继承的威力

12.2 建立继承

12.3 使用现有的对象

12.4 将相同类的对象存储到数组列表中

12.5 创建子类

12.6 总结

12.7 问与答

12.8 测验

12.9 练习

第13章 创建简单的用户界面

13.1 Swing和抽象窗口工具包

13.2 使用组件

13.3 创建自己的组件

13.4 总结

13.5 问与答

13.6 测验

13.7 练习

第14章 用户界面的布局

14.1 使用布局管理器

14.2 应用程序的界面布局

14.3 总结

14.4 问与答

14.5 测验

14.6 练习

第15章 响应用户输入

15.1 让程序监听

15.2 设置要监听的组件

15.3 处理用户事件

15.4 完善图形应用程序

15.5 总结

15.6 问与答

15.7 测验

[15.8 练习](#)

[第16章 创建复杂的用户界面](#)

[16.1 滑块](#)

[16.2 变更监听器](#)

[16.3 使用图像图标和工具栏](#)

[16.4 表](#)

[16.5 总结](#)

[16.6 问与答](#)

[16.7 测验](#)

[16.8 练习](#)

[第17章 在数据结构中存储对象](#)

[17.1 数组列表](#)

[17.2 哈希映射](#)

[17.3 总结](#)

[17.4 问与答](#)

[17.5 测验](#)

[17.6 练习](#)

[第18章 处理程序中的错误](#)

[18.1 异常](#)

[18.2 抛出和捕获异常](#)

[18.3 总结](#)

[18.4 问与答](#)

[18.5 测验](#)

[18.6 练习](#)

[第19章 创建线程程序](#)

[19.1 线程](#)

[19.2 使用线程](#)

[19.3 构造函数](#)

[19.4 在创建URL时捕获错误](#)

[19.5 启动线程](#)

[19.6 处理鼠标单击](#)

[19.7 循环显示链接](#)

[19.8 总结](#)

[19.9 问与答](#)

[19.10 测验](#)

[19.11 练习](#)

[第20章 使用内部类和闭包](#)

[20.1 内部类](#)

[20.2 闭包](#)

[20.3 总结](#)

[20.4 问与答](#)

[20.5 测验](#)

[20.6 练习](#)

[第21章 读写文件](#)

[21.1 流](#)

[21.2 将数据写入流中](#)

[21.3 读写配置属性](#)

[21.4 总结](#)

[21.5 问与答](#)

[21.6 测验](#)

[21.7 练习](#)

[第22章 利用JAX-WS开发Web服务](#)

[22.1 定义服务端点接口](#)

[22.2 创建服务实现Bean](#)

[22.3 发布Web服务](#)

[22.4 使用Web服务描述语言文件](#)

[22.5 创建Web服务客户端](#)

[22.6 总结](#)

[22.7 问与答](#)

[22.8 测验](#)

[22.9 练习](#)

[第23章 创建Java2D图形](#)

[23.1 使用Font类](#)

[23.2 使用Color类](#)

[23.3 创建自定义颜色](#)

[23.4 绘制直线和形状](#)

[23.5 绘制饼图](#)

[23.6 总结](#)

[23.7 问与答](#)

[23.8 测验](#)

[23.9 练习](#)

[第24章 编写Android app](#)

[24.1 Android简介](#)

[24.2 创建Android app](#)

[24.3 运行app](#)

[24.4 设计真实的app](#)

[24.5 总结](#)

[24.6 问与答](#)

[24.7 测验](#)

[24.8 练习](#)

[附录A 使用NetBeans IDE](#)

[A.1 安装NetBeans](#)

[A.2 创建新项目](#)

[A.3 创建新的Java类](#)

[A.4 运行应用程序](#)

[A.5 修复错误](#)

[附录B Java资源](#)

[B.1 可以考虑的其他图书](#)

[B.2 Oracle公司的Java官方站点](#)

[B.3 其他Java站点](#)

[附录C 本书站点](#)

[附录D 设置Android开发环境](#)

[D.1 起步](#)

[D.2 安装Eclipse](#)

[D.3 安装在Eclipse中使用的Android插件](#)

[D.4 设置你的手机](#)

[欢迎来到异步社区！](#)

版权信息

书名：Java入门经典（第7版）

ISBN：978-7-115-40036-9

本书由人民邮电出版社发行数字版。版权所有，侵权必究。

您购买的人民邮电出版社电子书仅供您个人使用，未经授权，不得以任何方式复制和传播本书内容。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。

• 著 [美] Rogers Cadenhead

译 郝记生

责任编辑 傅道坤

• 人民邮电出版社出版发行 北京市丰台区成寿寺路11号

邮编 100164 电子邮件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

- 读者服务热线: (010)81055410

反盗版热线: (010)81055315

版权声明

Rogers Cadenhead: Sams Teach Yourself Java in 24 Hours(7th Edition)

ISBN: 0672337029

Copyright © 2014 by Pearson Education, Inc.

Authorized translation from the English languages edition published by Pearson Education, Inc.

All rights reserved.

本书中文简体字版由美国Pearson公司授权人民邮电出版社出版。未经出版者书面许可，对本书任何部分不得以任何方式复制或抄袭。

版权所有，侵权必究。

内容提要

本书通过大量示例程序循序渐进地引导读者快速掌握使用Java开发程序的基本技能。

本书总共24章，先讲解了Java程序的编写流程、工作原理等内容；然后介绍了有关Java编程的基本知识，包括变量、条件语句、循环语句、数组和对象等内容；随后介绍了创建图形用户界面、使用内部类和闭包、读写文件，以及使用字体、颜色和图形等相关的知识。本书还介绍了如何使用Java来开发Android app。本书每章都提供了示例程序清单，并辅以示例输出和代码分析，以阐述该章介绍的主题。为加深读者对所学内容的理解，每章末尾都提供了常见问题及其答案以及练习和测验。

本书可作为初学者学习Java编程技术的教程，也可供其他语言的程序员学习Java时参考。

关于作者

Rogers Cadenhead是一名作家、计算机程序员、Web开发人员，他已经编写了20多本与Internet相关的图书，其中包括*Sams Teach Yourself Java in 21 Days*。他维护着Drudge Retort和其他站点，这些站点的年访问量有2000万人次。本书的官方站点是www.java24hours.com。

献辞

我从13岁起开始在Timex Sinclair 1000上编写程序，这是一台具有3.25MHz处理器、2KB存储的计算机，而且使用电视作为显示器。我要将本书献给我的父亲——Roger Cadenhead。他购买了这台计算机后，我很快就据为己有，但是我的父亲并没有为此抱怨。感谢我的父亲，是他带领着我走到了今天。

致谢

感谢Sams和Pearson的员工，尤其是Mark Taber、Tonya Simpson、Seth Kerney、Barbara Hacha和Boris Minkin。没有作者可以凭借一己之力来写出一本书。他们出色的工作给了我很多帮助。

谢谢我的妻子Mary和儿子Max、Eli、Sam。

前言

作为一名计算机图书作者，我花费了大量的时间呆在书店的计算机图书区，在假装阅读最新一期*In Touch Weekly* 杂志的同时，观察读者阅读图书的行为。

根据我的观察，读者拿起本书并翻到前言后，在他将书放下前往咖啡厅喝杯咖啡前，给我留下的时间大约只有13秒。

因此，这里长话短说：使用Java来进行计算机编程比想象的容易。我本不应该这样说，因为成千上万的程序员正是凭借其Java技能在软件开发、Web应用程序编程和移动app开发领域获得了高薪职位，对他们来说，最不想让老板知道的是，只要坚持不懈并有一点空闲时间，任何人都能学会当前最为流行的编程语言。通过阅读本书，你可以快速掌握Java编程。

任何人都能学会如何编写计算机程序，尽管他们不能对DVR进行编程。Java是最值得学习的编程语言之一，因为这种功能强大的实用技术已被全球大量的程序员采用。

本书是为非程序员、讨厌学习编程的程序员新手，以及经验丰富但想快速掌握Java的程序员编写的。本书使用的是该语言的最新版本——Java 8。

由于Java具有“让一切成为可能”的特性，因此成为一种非常流行的编程语言。你可以使用Java来创建具有图形用户界面的程序、设计充分利用Internet的软件、连接Web服务、开发在Android手机或平板上运行的游戏，等等。

本书引导读者从零开始学习Java编程，它以平实的语言阐述概念，并包含大量要求读者逐步创建的示例程序。读完本书，读者就能编写自己的Java程序，对自己使用该语言的能力充满信心，进而更深入地学习它；读者还将获得日益重要的技能，如网络计算、图形用户界面设计和面向对象编程。

就当前而言，这些术语对你来说也许并不重要。事实上，正是它们使得人们对编程充满了恐惧，并认为很难掌握。然而，如果你能使用计算机在Facebook上创建电子相簿、支付税款、处理电子表格，你就能通过阅读本书编写计算机程序。

By the Way

注意

现在，如果你还是愿意去喝咖啡而不是学习Java，请将本书放回书架，并将封面朝向书店中人来人往的通道，以方便别人找到它。

第1章 成为程序员

本章介绍如下内容：

- 找到学习Java的理由；
- 发现程序是如何工作的；
- 选择Java开发工具；
- 准备编写第一个程序。

你可能听说过计算机编程及其困难，它要求你获得计算机科学专业的学位，需要投入几千美元来购买计算机硬件和软件，需要敏锐的分析头脑，需要有耐心，而且对含咖啡因的饮料有强烈的爱好。

其实，在上述条件中，除喜欢喝含咖啡因的饮料外，其他所有条件都是不正确的。尽管多年以来计算机程序员一直都说编程很难，而且这种说法可以让程序员更容易找到高薪工作，但是，编程工作其实比大多数人想象的要容易。

现在是学习编程的最佳时代。我们可以从网上免费下载各种编程工具，数以千计的程序员以开源的形式发布他们编写的程序，以方便他人查看软件是如何编写的、修复软件中的错误，以及对程序做出改进。即使是在经济紧缩时期，许多公司也仍然在招聘程序开发人员。

现在是学习Java的最好时机，因为该语言无处不在。如今有数十亿计的移动设备使用的是Android操作系统，其上的app都是使用Java编写

的。如果你有一台Android手机，每当你搜索电影、通过网络电台收听摇滚乐，或者是玩“愤怒的小鸟”游戏时，你沉浸其中的就是Java程序员开发的软件。

本书旨在向三类人讲授Java编程知识：

1. 之前从没有尝试过编程的新手；
2. 尝试过编程但是又讨厌编程（如同伏地魔讨厌英国孤儿院的孩子一样）的初学者；
3. 懂其他编程语言但是希望迅速掌握Java的急性子。

为了实现这一目的，本书尽可能使用直白平实的语言而不是行话或者晦涩的首字母缩写，并对新出现的编程术语进行详细解释。

如果这种尝试得以成功，那么读者在读完本书后将获得丰富的编程技能，而这些技能对读者来说曾经是一种挑战。阅读本书后，读者将能够编写程序，更自信地参加编程课程，阅读编程图书，从而更轻松地学习新的编程语言（我的意思是“编程”语言。本书不会帮助你掌握西班牙语、世界语以及克林贡这种外星语言）。

当然，读者也会掌握Java这种应用最为广泛的编程语言。

本章将会简单介绍编程的概念，然后讲解如何设置计算机，以便用它编写和运行Java程序。

1.1 选择编程语言

如果你能够娴熟地使用计算机来准备一个漂亮的简历，或者是计算账目的收支平衡，亦或是将你在假期中拍摄的照片上传到Facebook上与他人共享，那么你就能在计算机上编写程序。在我曾经懵懂年少的年代，人们使用各种形式的BASIC语言来学习编写程序，原因是BASIC就是为初学者创建的。

By the Way

注意

BASIC语言的发明初衷是方便学生学习，BASIC中的字母B代表初学者（Beginner's）。使用BASIC语言的缺点是容易养成马虎的编程习惯。

如今，最流行的BASIC编程语言是Visual Basic.NET，它是微软开发的一种编程语言和一组开发工具，已经远远超出了BASIC的初衷。它又名VB.NET，旨在创建能在Windows操作系统的计算机和移动设备上运行的程序。另外一个流行的语言是PHP，这是一个用来创建站点的脚本语言。你可能听说过的其他广泛使用的语言有C++、C#、Ruby和Python。

这些编程语言都有其拥趸，但是在高中和大学的计算机科学课堂上讲授最为广泛的是Java。

相较于VB.NET和PHP等某些语言，来自Oracle的Java编程语言要更难学习，但是基于多个原因，将其作为入门语言仍然是不错的选择。学习Java的一个优势是你可以跨操作系统和计算环境来使用Java。Java程序可以是桌面软件、Web应用程序、Internet服务器、Android app，

可以运行在Windows、Mac、Linux和其他操作系统上。Java语言的这种通用性在早期被雄心勃勃的Java口号宣称为“一次编写，到处运行”。

By the Way

注意

早期的Java评论家都有一个不讨人喜欢的口号“一次编写，到处调试”。Java语言在1996年发布了第一个版本之后，走了很长的一段路才发展起来。

Java语言另外一个重要的优势是，你需要一个高度组织化的方法，才能让程序运行起来。如何编写程序，以及程序如何存储和修改数据，对这些东西你一点也不得马虎。

刚开始编写Java程序时，你可能不会将Java的这种挑剔行为当做优势。编写完程序后，你可能要修改多处错误才能让程序运行起来，从而对这种程序编写方式产生厌烦心理。

在接下来的几章中，你将学习Java的这些规则以及应避开的陷阱。

Java是由加拿大的计算机科学家James Gosling发明的，旨在提供一种更好的计算机编程方式。Gosling于1991年在Sun公司工作时，他对C++编程语言在项目上的执行方式颇为不满，因此决定创建一种更能胜任该工作的新语言。当然，尽管在Java是否优于其他编程语言这一点上仍然存在激烈的争论，但是它的成功已经证明了其最初的设计优势。世界上30亿台设备运行的是Java程序。这是一个相当惊人的数字。从

Java面世到现在，已经出版了1000多种与它相关的图书（这是我的第18本Java图书）。

不管Java是不是最好的语言，但绝对是值得学习的一种伟大的编程语言。在第2章，读者将有机会尝试编写Java程序。

学习任何一种编程语言后，再学习其他编程语言将会比较容易。很多语言彼此相似，因此在学习新语言时，不用完全从头开始。例如，很多 C++或 Smalltalk 程序员发现学习 Java相当容易，因为Java从这些早期的编程语言中借鉴了许多思想。同样，C#编程语言也从Java借鉴了很多思想，所以Java程序员也很容易就能学会C#编程语言。

By the Way

注意

本章多次提到了C++，也许读者不明白它指的是什么，以及它是如何发音的。C++可以读成“C加加”，它是由贝尔实验室的丹麦计算机科学家 Bjarne Stroustrup开发的一种编程语言。C++是C语言的增强版，这就是C++中“++”的含义。那为什么不叫C+呢？本书后面将介绍，++是一个计算机编程笑话。

1.2 告诉计算机做什么

计算机程序也叫软件，它告诉计算机该执行什么任务。计算机执行的任何操作（从启动到关机）都是由程序控制的。Mac OS X是程序，Minecraft是程序，控制打印机的驱动软件是程序，甚至Windows PC在崩溃时出现的蓝屏死机也是程序。

计算机程序由一系列命令组成，程序运行时，计算机按特定顺序处理这些命令。其中的命令称为语句。

如果你的家里雇佣了一名管家，而且你是一个具有A型人格的控制狂，你需要为管家制定一组详细的命令供其每日遵循，如下所示。

亲爱的Jeeves先生，在我外出向国会请求紧急援助之时，请为我照顾这些差事：

1. 用吸尘器打扫房间；
2. 去商场；
3. 买些酱油、芥末，并尽可能多买些加州寿司卷；
4. 回家。

谢谢！

Bertie Wooster

如果你告诉管家做什么，那么他就会灵活地完成你安排的任务：如果加州寿司卷卖没了，你的管家会给你买波士顿卷。

但是计算机没有回旋余地。它会按照你编写的程序严格执行，一次执行一个语句。

下面是一个采用BASIC语言编写的计算机程序示例，只有三行。现在我们来看一下这个程序，请读者不要纠结于该程序每一行所代表的意思。

```
1. PRINT "Hey Tom, it's Bob from the office down the hall."  
2. PRINT "It's good to see you buddy, how've you been?"  
3. INPUT A$
```

将该程序翻译为自然语言，就相当于让计算机来执行如下事情：

亲爱的个人计算机，

1. 请显示消息“Hey Tom, it's Bob from the office down the hall.”
2. 回答问题“It's good to see you buddy, how've you been?”
3. 给用户机会来回答这个问题。

你的主人Ima Coder

计算机程序中的每一行称为一条语句。计算机按特定顺序处理程序中的每条语句，就像厨师按菜谱炒菜，或管家Jeeves先生按照Bertie Wooster的指示行事那样。在BASIC语言中，行号用于标识语句的顺序。其他语言（如Java）不使用行号，而是使用不同的方式告诉计算机如何运行程序。

图1.1在ReadyBASIC中显示了一个三行的BASIC程序，ReadyBASIC是一个BASIC解释器，它是在www.readybasic.com站点上提供的。

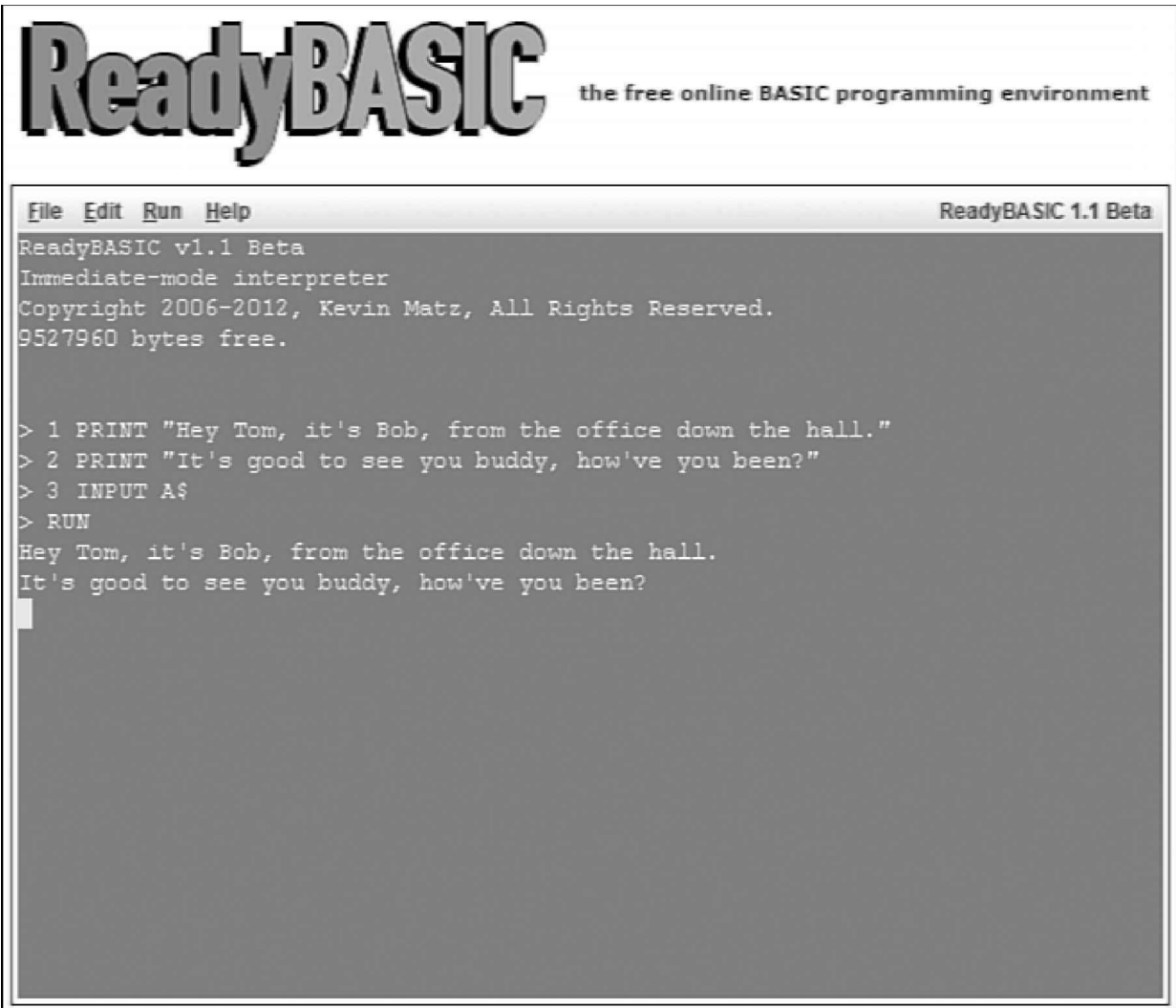


图1.1 一个BASIC程序示例

鉴于程序的运行方式，如果程序运行时发生错误，不能责怪计算机。毕竟，计算机只是严格按照你的指令去做。因此程序的运行错误应该归咎于程序员。

这真是一个坏消息。当然，好消息是，你不会对计算机造成永久伤害。在你学习使用Java编程时，没有计算机会受伤。

By the Way

注意

ReadyBASIC解释器是使用Java语言创建的。开发人员Kevin Matz创建了一个Java applet（一种在Web页面内运行的Java程序），用来解释BASIC代码的语句。

要想让ReadyBASIC工作，你的浏览器必须能够运行Java程序。如果有问题，可以访问Oracle的Java站点（www.java.com）来测试浏览器是否支持Java——如果有必要，请安装Java插件。

你可能也需要调整Java安全设置，以便程序运行。在Windows中，打开控制面板，单击Java，然后单击“安全”选项卡 [1]。你可以修改安全级别或者允许特定的网站运行Java applet。

1.3 程序的工作原理

构成计算机程序的语句集合称为源代码。

大多数计算机程序的编写方式都相同，就像写电子邮件一样：在文本窗口中输入每条语句。有些编程工具自带了源代码编辑器，而有些编程工具则可以与任何文本编辑软件一起使用。

编写好计算机程序后，将文件存盘。计算机程序通常有它们自己的扩展名，用于指出所属的文件类型。Java程序的扩展名为.java，如Calculator.java。

为了运行保存为文件的程序，你需要某些帮助。所需要的帮助类型取决于你使用的编程语言。有些语言需要解释器来运行程序。解释

器检查计算机程序的每一行，然后执行该行，然后再检查下一行。
BASIC的许多版本都是解释型语言。

解释型语言的最大优势是可以快速进行测试。当编写**BASIC**程序时，你可以立即进行测试，修复错误并再次测试。解释型语言的缺点是运行速度比其他程序慢。每一行语句需要先翻译成计算机可以运行的指令，而且一次只能翻译一行。

其他编程语言需要编译器。编译器接受一个程序，然后将其翻译成计算机能够理解的格式。它还能够使程序尽可能高效地运行。编译后的程序不需要解释器就可直接运行。

编译后的程序，运行速度要比解释型程序快，但是它需要花费一些时间来测试。你需要先编写程序，然后将其编译，之后才能尝试运行。如果你发现了错误并修复之后，必须再次编译该程序。

Java与众不同，它同时需要编译器和解释器。编译器将构成程序的语句转换成解释器可以运行的字节码。这里的解释器称之为**Java**虚拟机。

Java虚拟机也称为**JVM**，它可以使得相同的**Java**程序在无需修改的情况下，就能在不同的操作系统和不同的计算设备上运行。虚拟机将字节码转换成设备的操作系统可以执行的指令。

后面编写**Java**程序时，读者将更详细地了解这一点。



By the
Way

注意

如果你最喜欢的文本编辑器是一款字处理程序，能够支持粗体、字号或其他风格的字体，则在编写计算机程序时要么选择另外一款编辑器，要么不要使用这些功能。程序应该是没有任何特殊格式的文本文件。

Windows 操作系统自带的文本编辑器 Notepad 将所有文件都保存为无格式的文本。你也可以使用Mac上的TextEdit或者Linux系统上的vi或Emacs编辑器来创建无格式的文本文件。

1.4 为什么程序不能正常工作

很多编程新手在测试程序时感到气馁，因为到处都是错误。有些是语法错误，当计算机在查看程序时，被语句的编写方式给搞糊涂了，这样的语句就会被识别为语法错误。还有一些错误是逻辑错误，这类错误只能在程序员测试程序时发现（也有可能完全被忽略掉）。逻辑错误通常会导致计算机执行意想不到的操作。

当你开始编写自己的程序时，将会熟悉各种错误，这是必不可少的环节。编程错误称为 **bug**，该术语起源于100年前甚至更早，用于描述技术设备的错误。

修复错误的过程也有自己的术语：调试（**debugging**）。

现在有多种方法可以用来描述编程错误，而且这并不是一个巧合。不论是否愿意，在学习编写程序的过程中，都将获得丰富的调试经验。

**By the
Way**

注意

第一个计算机bug于1997年被美国计算机科学家Grace Hopper所在的团队发现。Hopper正在Harvard测试计算机时，发生了继电器故障。发生故障的原因被证实不是由软件引起的，而是由真正的虫子（bug）引起的。她将死去的飞蛾拿掉后开始调试计算机，并在日志中记录到“发现的第一例真实的bug事件（first actual case of bug being found）”。

1.5 选择一个Java编程工具

在开始编写Java程序前，你必须有一个Java编程工具。当前有多种Java编程软件，其中包括简单的Java Development Kit，还包括稍微复杂的Eclipse、IntelliJ IDEA和NetBeans。后3种工具都是集成的开发环境（IDE），它们是专业的程序员用来完成工作的强大工具。

当Oracle发布新的Java版本时，对其进行支持的第一款工具是Java Development Kit（JDK）。

为了创建本书中的程序，读者必须使用JDK版本8，或者是可以在JDK版本8上工作的编程工具。JDK是一组用来创建Java软件的免费的命令行工具。JDK不具备图形用户界面，如果你之前没有在诸如Windows命令提示符或Linux命令行界面这样的非图形界面环境中工作过，那么当你使用JDK时将会感觉很有挑战性。

NetBeans是Oracle 免费提供的另外一个工具，相比于JDK，它可以更容易地编写和测试Java代码。NetBeans提供了图形用户界面、源代码编辑器、用户界面设计器，以及项目管理器等功能。它可以与运行在

后台的JDK形成互补，因此在开始编写Java程序时，必须同时在系统上安装了这两种工具。

本书中的程序是用NetBeans编写的，用户可以下载NetBeans，并与JDK分开安装。你也可以使用其他Java工具，只要他们支持JDK 8。

By the Way

注意

在本书中你不一定非要使用NetBeans。你也可以使用JDK或其他工具来创建、编译和运行程序（这也是大多数项目所需要进行的工作）。之所以在本书中使用NetBeans，是因为本书之前版本的读者已经证明NetBeans要比JDK容易。我所有的Java程序都是使用NetBeans编写的。

1.6 安装Java开发工具

本书每章都以一个Java编程项目收尾，你可以通过该项目来加深对相关主题知识的理解。

如果你的计算机上没有安装Java编程工具，也就不能进行Java编程。

如果你已经安装了支持Java的工具，你可以用它来开发本书后面23章中的示例程序。然而，你应熟悉如何使用这些工具，因为同时学习如何使用Java和复杂的IDE可能令人畏惧。

在你阅读本书时，推荐用来编程的工具是**NetBeans 8.0**，用户可以从Oracle的站点<http://netbeans.org> 免费下载该工具。尽管用户需要花些时间来学习**NetBeans**的高级特性，但是它可以用来轻松创建和运行简单的**Java**应用程序。

有关下载和安装**NetBeans**的方法，请阅读附录A。

1.7 总结

本章介绍了有关计算机编程的概念：提供一组称为语句的指令，告诉计算机做什么。本章也介绍了选择**Java**而非其他编程语言学习编程的原因。

你也可以下载并安装**Java**开发工具，以用来编写本书所有的示例程序。

如果你问10位程序员最好的编程语言是什么，没准会得到10个答案，与之而来的还有“我使用的编程语言能甩你的编程语言几条街”等嘲讽以及“你的源代码臃肿不堪”等笑话。在这样的争论中，**Java**受到了一致好评，原因是它的设计很巧妙，而且非常灵活，并被广泛采用。

如果读者仍不清楚程序、编程语言或**Java**等概念，也不要气馁。下一章将详细介绍**Java**程序的创建过程，届时，读者将会明确上述这些概念。

1.8 问与答

问：BASIC、C++、Smalltalk、Java等语言的名字是什么意思？

答：BASIC是Beginner's All Symbolic Instruction Code的首字母的缩写。C++是为了改进C语言而创建的编程语言，而C语言是B编程语言的改进版。Smalltalk是一种革命性的面向对象的编程语言，开发于20世纪70年代，而且它的很多理念都被Java所采用。

Java没有采用传统语言的命名方式：使用首字母缩写或其他有意义的术语。Java是Java开发者最喜欢的名称，它击败WebRunner、Silk、Ruby等名称脱颖而出（当时Ruby编程语言还不存在）。

当我创建属于我的第一个编程语言时，会将它命名为Salsa——因为每个人都喜欢它嘛！

问：为什么解释型语言的速度比编译型语言慢？

答：原因与现场翻译的速度比翻译书面演讲稿慢相同。现场翻译时，翻译人员需要思考刚刚讲过的每句话，然而其他翻译人员可将演讲作为一个整体，并采取优化的方式以提高翻译速度。编译型语言比解释型语言快得多，是因为可以采取提高程序的运行效率。

1.9 测验

通过回答下列问题来检测对本章介绍的知识的掌握程度。

1.9.1 问题

1. 人们认为计算机编程很难，下面哪个不是其原因？

- a. 程序员散布该谣言，以提升就业前景。
 - b. 到处充斥行话和首字母缩写。
 - c. 认为编程很难学习的人可以有资格获得政府的援助。
2. 下面哪种工具以每次一行的方式运行计算机程序？
- a. 缓慢的工具。
 - b. 解释器。
 - c. 编译器。
3. James Gosling为什么要发明Java？
- a. 他对在一个项目中使用的语言不满意。
 - b. 他的摇滚乐队当时没有任何演出。
 - c. 如果在工作期间不能浏览YouTube，Internet将很单调。

1.9.2 答案

- 1. c. 计算机图书的作者也没有得到政府的援助。
- 2. b. 解释器每次解释一行。编译器事先理解指令，所以程序运行起来更快。
- 3. a. 他对C++感到失望。在他创建Java的1991年，YouTube还没有诞生。

1.10 练习

如果想更多地了解关于Java和计算机编程的信息，可完成下列练习。

- 要了解程序员选择Java语言的原因，可以访问Stack Overflow问答网站（www.stackoverflow.com/questions/209555），查看人们是如何回答“Why would you choose the Java Programming language over others（你为什么在众多编程语言中选择了Java）”这个问题的。
- 使用普通语言编写一组指令，将温度由摄氏度转换为华氏度。将这些指令分为多个短句子，争取一行一个语句。

要获悉每章末尾的习题答案，请访问本书的配套网址：

www.java24hours.com。

[1] 译者注：控制面板中找不到相应的设置，反而是通过打开IE浏览器，然后单击菜单栏中“工具”菜单下的“Internet选项”，从中可以找到“安全”选项卡，并完成相应设置。

第2章 编写第一个程序

本章介绍如下内容：

- 在文本编辑器中输入一个Java程序；
- 使用括号组织程序；
- 将信息存储到变量中；
- 显示存储在变量中的信息；
- 保存、编译和运行程序。

第1章已经提到，计算机程序是一组告诉计算机做什么的指令。这些指令使用编程语言输入到计算机中。

在本章，读者将通过将指令输入到文本编辑器的方式来创建第一个Java程序。输入完毕之后，可以保存、编译并测试该程序。然后你可以破坏该程序，然后再进行修复。

2.1 编写程序所需的工具

第1章讲到，要创建Java程序，你必须有支持Java Development Kit (JDK) 的开发工具，比如NetBeans集成开发环境 (IDE)。你还需要可以编译和运行Java程序的工具，以及用来编写Java程序的文本编辑器。

对于大多数编程语言来说，计算机程序都是通过文本编辑器（也称为源代码编辑器）中输入文本的方式来编写的。有些编程语言自带了文本编辑器。**NetBeans**就包含了用来编写Java程序的编辑器。

Java程序是简单的文本文件，没有诸如文本居中、粗体等其他特殊格式。**NetBeans**源代码编辑器很像一个功能增强了的简单文本编辑器。当你输入文本来标识编程语言的不同元素时，文本将变成不同的颜色。**NetBeans**本身也带有适当的缩进格式，而且编辑器内也提供了有用的编程文档。

由于Java程序是文本文件，因此可以使用任何文本编辑器打开Java程序，并对其进行编辑。你也可以使用**NetBeans**编写Java程序，然后在Windows Notepad（记事本）中打开它，并做出相应的修改，然后再在**NetBeans**中打开该程序，这不会造成任何问题。

2.2 创建Saluton程序

你要创建的第一个Java程序将显示计算机科学界的传统问候语“Saluton mondo！”。

在**NetBeans**中输入第一个程序之前，先采取如下步骤来创建一个称为Java24的新项目。

1. 选择菜单命令File->New Project，打开New Project对话框。
2. 选择项目分类“Java”和项目类型“Java Application”，然后单击Next按钮。

3. 输入“Java24”作为该项目的名称（如果之前已经创建了该项目，则会看到错误信息“Project folder already exists and is not empty”）。

4. 取消选中“Create Main Class”复选框。

5. 单击Finish按钮。

Java24项目在其文件夹中建立起来。在阅读本书的过程中，你可以为编写的所有Java程序使用这个项目。

2.3 开始输入程序

NetBeans将所有相关的程序分组列入到一个项目中。如果你还没有打开Java24项目，可采用如下方式打开。

- 选择菜单命令File->Open Project。
- 找到并选择NetBeansProjects文件夹（如果有必要）。
- 选择Java24，并单击Open Project按钮。

Java24项目出现在项目（Projects）面板中，它靠近一个咖啡杯图标，而且有一个加号（+），点击这个加号可以查看项目包含的文件和文件夹。

为了添加一个新的Java程序到当前的项目中，选择File->New File，打开New File Wizard对话框，如图2.1所示。

分类（Categories）面板会列出用户可以创建的所有 Java 程序的类型。在该面板中单击“Java”文件夹来查看属于该分类的文件类型。对本项目而言，选择“Empty Java File”类型，然后单击Next按钮。

出现一个New Empty Java File对话框。根据下述步骤来编写程序。

1. 在Class Name字段，输入“Saluton”，
2. 在Package字段，输入“com.java24hours”。
3. 然后单击Finish按钮。

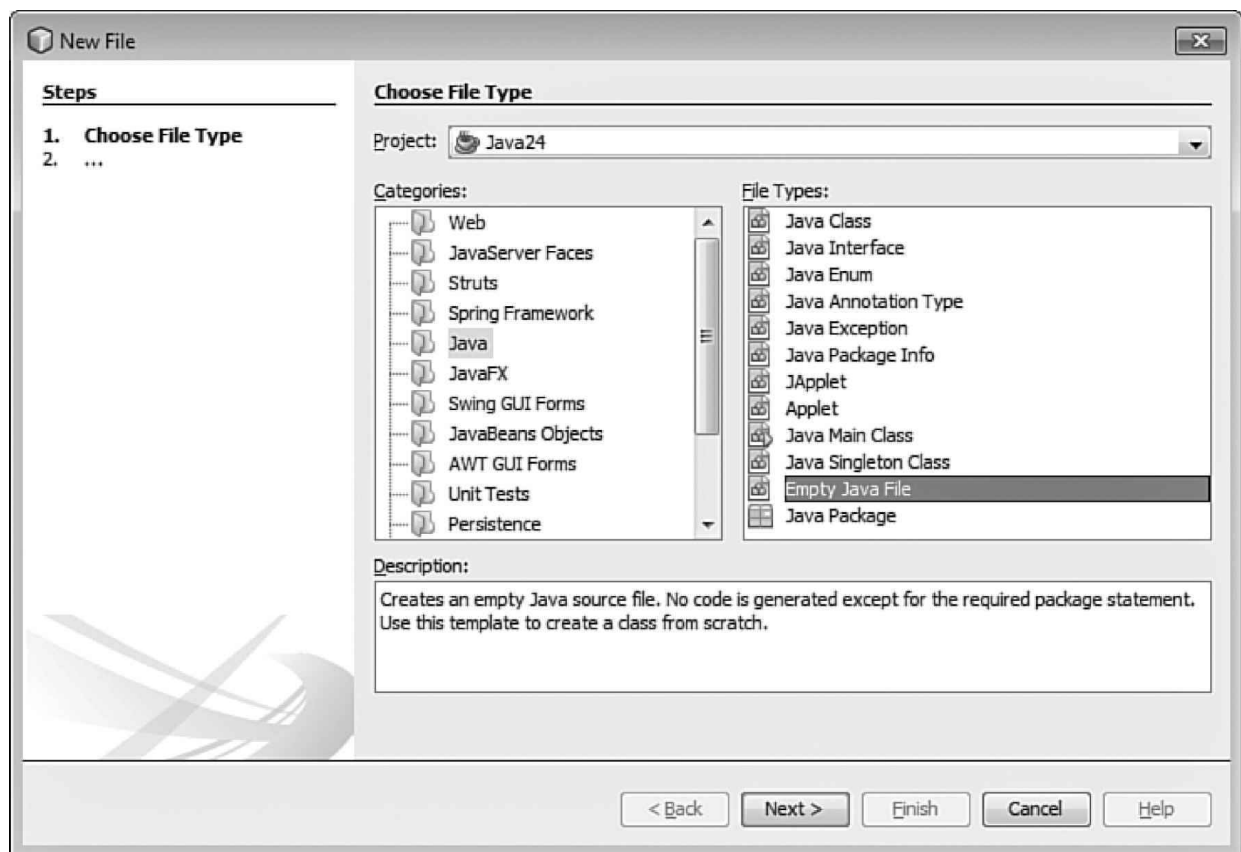


图2.1 New File Wizard对话框

现在可以开始编程工作。一个名为**Saluton.java**的空文件将在源代码编辑器中打开。在源代码编辑器中，通过输入程序清单2.1中的每一行语句，由此开启你的**Java**编程生涯。这些语句称为程序的源代码。

Watch Out!

警告：

不要输入每行前面的行号和冒号，它们在本书中的目的是方便引用特定的代码行。

程序清单2.1 Saluton程序

```
1: package com.java24hours;  
2:  
3: class Saluton {  
4:     public static void main(string[] arguments) {  
5:         // My first Java program goes here  
6:     }  
7: }
```

确保代码的大小写与该程序清单中一致，并使用空格键或Tab键在第4行～第6行前面插入空白。当输入完后，选择**File->Save**，保存文件。

当前，**Saluton.java**只包含**Java**程序的架构。读者可以创建多个开头与此相同的程序，当然，第3行的单词**Saluton**除外。该单词表示程序的

名称，而且每个程序都不同。第5行是一个英语句子，你应该能看得懂。其他内容对你来说是全新的。

2.3.1 class语句

程序的第1行如下：

```
package com.java24hours;
```

包（package）是将Java程序组合起来的一种方式。这一行告诉计算机将程序的包名称命名为com.java24hours。

第2行是一个空行，第3行如下所示：

```
class Saluton {
```

把这句话翻译成自然语言的意思是：计算机，请将我的Java程序命名为Saluton。

第1章讲到，输入到计算机的每条指令都被称为语句。class语句让你能够给计算机程序指定名称，它还可以确定程序的其他方面，这将在后面介绍。术语 class 的意义在于 Java程序也叫class（类）。

在这个例子中，程序名**Saluton**与文档名**Saluton.java**匹配。Java程序的名称应与其文件名的第1部分相同，而且大小写完全一致。

如果程序名与文件名不匹配，编译有些Java程序时可能会出错，而是否出错则取决于如何使用**class**语句来配置程序。

2.3.2 **main**语句的作用

该程序的下一行如下：

```
public static void main(String[] arguments) {
```

这行语句告诉计算机：程序的主要部分从这里开始。Java程序被组织成多个不同的部分，所以在运行程序时，需要有方法指出首先执行哪部分。

大多数Java程序的入口都是**main**语句，但**applet**、**servlet**和**app**例外。其中，**applet**是由Web浏览器在Web页面上运行的程序；**servlet**是由Web服务器运行的程序；而**app**是由移动设备运行的程序。

下来的几章中，你编写的大多数程序都将**main**作为起点。原因是你是在计算机上直接运行它们，而**applet**、**app**和**servlet**则是由其他程序或设备间接运行的。

为了与其他类型进行区分，我们将直接运行的程序称为应用程序。

2.3.3 大括号

在Saluton程序中，第3、4、6、和7行都包含一个大括号，要么是“{”，要么是“}”。这些大括号用于将程序中的语句分组（与小括号类似，小括号是用来将句子中的单词分组）。在左大括号“{”和右大括号“}”之间的内容属于同一组。

这些语句组称为块。在程序清单2.1中，第3行的左大括号“{”与第7行的右大括号“}”对应，它们将整个程序作为一个块。可以用这种方式来指示程序的开始和结尾。

块可以包含其他块（就像小括号可以这样使用“（（.....））”一样）。在Saluton程序中，第4行和第6行的大括号指定了另一个块。

这个块以main语句打头。程序运行时，计算机将运行main语句块中的内容。

下面的语句是该语句块中唯一的内容：

```
// My first Java program goes here
```

该行这是占位行，行首的//告诉计算机忽略本行，在程序中放置它的目的在于方便人们阅读程序的源代码。用于该目的的行被称为注释。

By the Way

注意

NetBeans可以显示块的开始位置和结束位置。在Saluton程序的源代码中单击其中一个大括号，这个大括号以及与之对应的大括号将显示为黄色。封装在这些黄色大括号中的Java语句组成了一个块。在Saluton这样的小程序中，这一提示的用处不大，但是当程序很长时，它可以免得让你像个无头苍蝇一样进行查找。

至此，你已经编写完了一个完整的Java程序。你可以编译它，但是运行它时什么也不会发生。因为你没有告诉计算机要做什么。main语句块只包含了一行注释，而它将被计算机忽略。你必须在main语句块的左大括号和右大括号之间添加一些语句。

2.4 在变量中存储信息

在编写的程序中，你需要将信息临时存储在某个地方，为此可以使用变量。变量是可存储整数、浮点数、真假值、字符及文本行等信息的地方。在变量中存储的信息可以修改，“变量”因此而得名。

在Saluton.java文件中，用如下语句来替换第5行：

```
String greeting = "Saluton mondo!";
```


这条语句告诉计算机：将文本“**Saluton mondo!**”存储到变量 **greeting** 中。

在Java程序中，必须告诉计算机变量存储的信息类型。在这个程序中，**greeting**变量是一个字符串—可以包含字母、数字、标点符号及其他字符的文本行。在该变量前面添加**String**之后，就可以用来存储字符串值。

在程序中输入这条语句时，必须在行尾输入分号（**;**），在Java程序中，每条语句都要以分号结束。这就像句尾的句点，计算机使用它们来判断一条语句的结束位置以及下一条语句的开始位置。

对我们人类来说，每一行只输入一个语句的方式，可以使程序更容易理解。

显示变量的内容

如果此时运行程序，将不会显示任何内容。在**greeting**变量中存储文本行的命令在幕后运行。要让计算机显示它在做什么，则需要显示该变量的内容。

在**Saluton**程序中，在语句**String greeting = "Saluton mondo!";**的后面插入一个空行，然后输入下面的语句：

```
System.out.println(greeting);
```

这条语句告诉计算机显示存储在`greeting`变量中的值。语句`System.out.println`告诉计算机在系统输出设备上（也就是你的显示器）显示信息。

现在计算机可以显示信息了。

2.5 保存编写好的程序

程序现在应如程序清单2.2所示，当然，可能第5行和第6行使用的间距略有不同。进行必要的改正，然后存储该文件（通过选择File->Save来执行）。

程序清单2.2 Saluton程序的完整版本

```
1: package com.java24hours;
2:
3: class Saluton {
4:     public static void main(String[] arguments) {
5:         String greeting = "Saluton mondo!";
6:         System.out.println(greeting);
7:     }
8: }
```

计算机运行该程序时，将运行`main`语句块中的第5行和第6行。用自然语言而不是Java来编写该程序时，将如程序清单2.3所示。

程序清单2.3 Saluton程序的逐行分解

```
1: Put this program in the com.java24hours package.
```

```
2:
3: The Saluton program begins here:
4:     The main part of the program begins here:
5:         Store the text "Saluton mondo!" in a String variable
named
        ➔ greeting
6:         Display the contents of the variable greeting
7:     The main part of the program ends here.
8: The Saluton program ends here.
```

程序清单2.4所示为使用克林贡语言编写的程序，该语言由电影《星际迷航》中的外星种族所使用。

程序清单2.4 使用克林贡语言编写的Saluton程序

```
1: This program belongs to the house of com.java2hours!
2:
3: Begin the Saluton program here if you know what's good for you!
4:     The main part of the program begins here with honor!
5:         Store the gibberish "Saluton mondo!" in a String
variable
        ➔ called greeting!
6:         Display this gibberish from a tongue inferior to
Klingon!
7:     End the main part of the program here to avoid my wrath!
8: End the Saluton program now and be grateful you were spared!
```

2.6 将程序编译为class文件

在运行Java程序之前，必须先编译它。在编译程序时，输入到计算机中的程序指令被转换为计算机可以更容易理解的一种形式。

在NetBeans中，程序在保存时会自动进行编译。如果你输入程序清单2.2中的指令，则该程序将顺利通过编译。

该程序在编译之后生成一个新的文件，名为Saluton.class。所有的Java程序都将编译为类（class）文件，且其文件后缀名为.class。Java程序也可以由协同工作的多个类文件组成，但是对于Saluton这样简单的程序来说，只需要一个类文件。

编译器将Java源代码转换为字节码，这是一种Java虚拟机（JVM）可以运行的格式。

By the Way

注意

只有Java程序在编译出现错误时，Java编译器才会有提示。如果你成功地编译完了一个程序，期间没有任何错误，则Java编译器不会有任何“动静”。这有点虎头蛇尾。当我在刚开始学习Java编程时，我曾经希望在编译取得成功时，能够响起庆祝的号角。

2.7 修复错误

当在NetBeans源代码编辑器中编译程序时，如果出现错误，在编辑面板中对应行的左边会出现一个红色警示图标，如图2.2所示。

错误图标

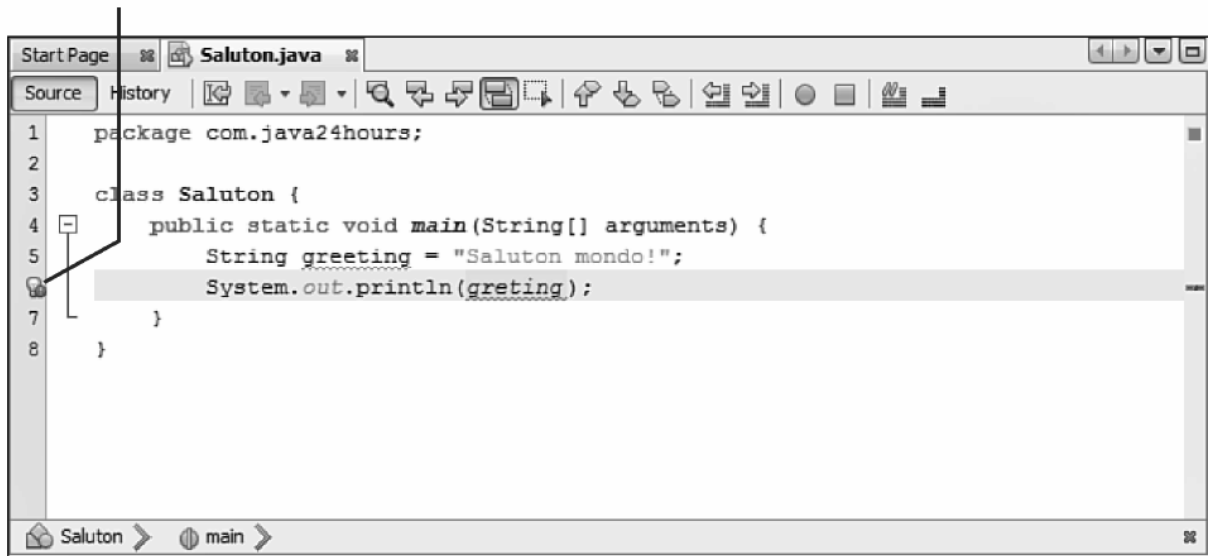


图2.2 在源代码编辑器中定位错误

这个图标显示在触发错误的那一行代码上。可以单击该图标来显示该错误信息，该错误信息会用如下细节来解释这个编译器错误。

- Java程序的名字。
- 错误的类型。
- 出错的行号。

比如，在编译Saluton程序时，可能会看到如下错误信息。

```
cannot find symbol.
symbol : variable greting
location: class Saluton
```

这里的错误消息是第一行“cannot find symbol”。很多编程新手往往会对这些消息感到迷惑。当读者不明白错误消息的意思时，不要为此枉费时间。只要看一下错误所在的行数，然后寻找最明显的原因即可。

Did you Know?

提示

本书官方站点www.java24hours.com包含了本书所有程序的源文件。如果你在Saluton程序中找不到任何输入错误以及其他错误原因，但是错误仍然存在，则可以去本书的官方站点下载Saluton.java程序，然后再运行该程序。

例如，你能确定下面的语句哪里出错了么？

```
System.out.println(greting);
```

这里的错误是变量名输入错误，即变量名应该是greeting，而不是greting（故意在NetBeans中添加该输入错误，看看会发生什么）。

在创建Saluton程序时，如果提示有错误消息，可以检查你的程序是否与程序清单2.2中的相同，然后再修改你找到的不同之处。要保证每一个字母的大小写都是正确的，而且也包含了所有的标点符号（例如{、}和;）。

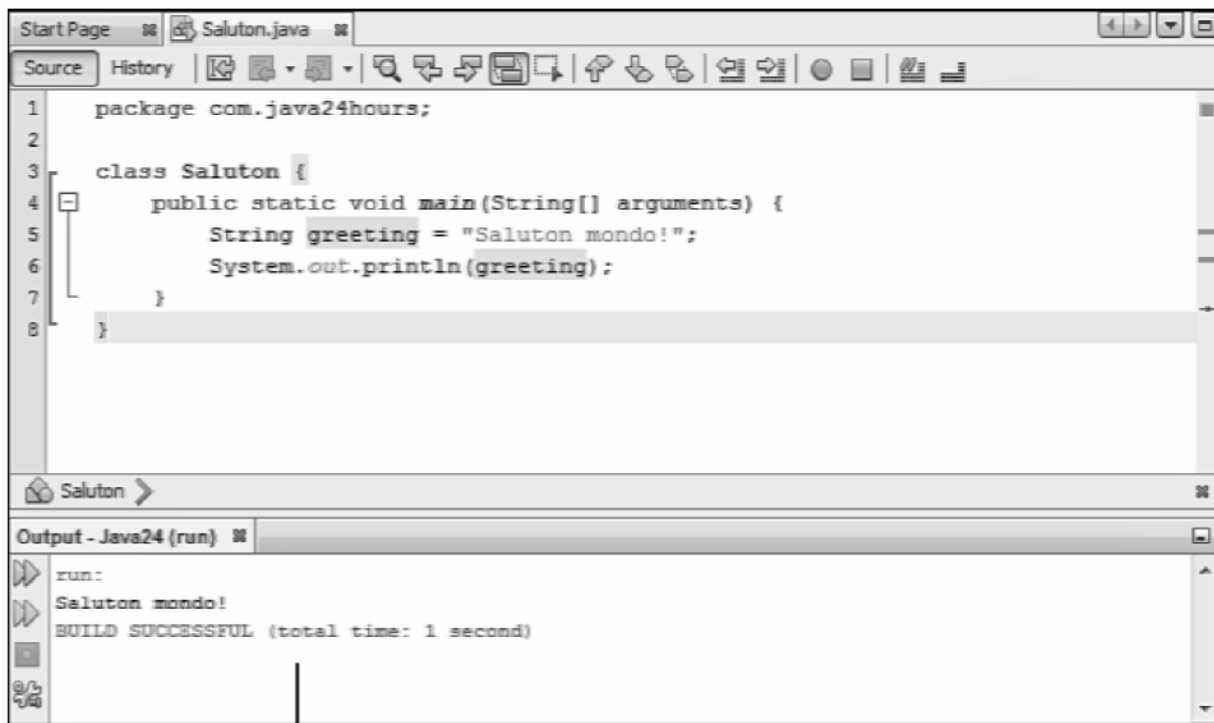
通常情况下，仔细检查错误消息指示的那一行，足以发现需要修复的所有错误。

2.8 运行Java程序

要查看Saluton程序的结果是否如你所愿，可使用Java虚拟机（JVM）运行类文件，JVM就是运行所有Java代码的解释器。在NetBeans中，选择菜单命令Run->Run File。在源代码编辑器的下面将会打开输出面板。如果没有错误，则该程序会在该面板中显示输出结果，如图2.3所示。

如果你看到了在Java程序中输入的文本“Saluton Mondo!”，那么你的计算机也就对世界发出了问候—这是计算机编程领域的一个传统，这对于很多程序员来说就像咖啡、短袖衬衫和Call of Duty（使命召唤）一样重要。

读者可能会问为什么“Saluton mondo!”是一句传统的问候语呢。该短语在世界语中的意思是“Hello world! ”，而世界语是由Ludwig Zamenhof在1887年创建的一种人工语言，用于促进国际间的交流。我之所以使用它，是因为它只是一个传统的问候语。



输出面板

图2.3 运行你的第一个Java程序

By the Way

注意

Oracle在网上为Java语言提供了全面的文档。在阅读本书时，你不需要这个文档，因为这个文档在引入每一个主题时，讨论的太过详细。当你想扩大知识量以及编写程序时，这个文档就排上用场了。

这个文档可以下载，但是在Oracle的网站上在线浏览所需内容无疑更为方便。读者可以通过<http://download.java.net/jdk8/docs/api> 下载最新的Java文档。

2.9 总结

在本章，读者第一次创建了一个Java程序，并学会了开发Java程序所需要的3个基本步骤。

1. 使用文本编辑器或NetBeans这样的工具编写程序。
2. 将程序编译为类文件。
3. 让JVM运行该类文件。

与此同时，读者还学习了一些基本的计算机编程概念，如编译器、解释器、块、语句和变量。随着后续章节的学习，读者将会对这些概念越来越清晰。只要读者在本章成功地运行了Saluton程序，就可以进入下一章。

2.10 问与答

问：在Java程序的每行中插入适当数量的空格有多重要？

答：对计算机而言，这完全不重要。空格无疑会让阅读计算机程序的人受益，但Java编译器对空格的数量并不关心。在编写Saluton程序时，你也可以不使用空格或Tab键进行缩进，而且它也能成功编译。

虽然每行开头的空格数不重要，但在Java程序中应采用一致的间距和缩进方式。原因是空格有助于查看程序的组织结构以及语句所属的程序块。

你编写的程序对其他程序员（包括你自己）来说必须是可理解的。当几周或几个月后，你需要修复bug或进行改进的时候，必须能够看得懂代码。间距和缩进的一致性编程风格的一部分。优秀的程序员会采用一种风格并在他们所有的代码进行体现。

问：Java程序被描述为一个类和一组类。哪种说法是正确的？

答：两者都正确。在接下来几章中，你创建的简单Java程序将被编译为扩展名为.class的单个文件，你可以使用JVM来运行它们。Java程序也可以由一组协同工作的类组成。该主题将在第10章详细介绍。

问：既然每条语句都必须以分号结尾，为何注释行“//My first Java program goes here”不需要以分号结尾？

答：编译器会完全忽略掉注释行。如果在程序中加入“//”，则是告诉Java编译器忽略该行中“//”右边的所有内容。下面的例子演示了如何在语句所在的行添加注释：

```
System.out.println(greeting); // hello, world!
```

问：我在编译器指出有错误的行中没有发现任何错误，我应该怎么办？

答：错误消息中显示的行号并不总是程序中发生错误的地方。检查错误消息指出的行号上面的行，看看能否找到拼写错误或其他bug。错误通常位于同一个程序块中。

2.11 测验

通过回答下列问题检测对本章介绍的知识的掌握程度。

2.11.1 问题

1. 编译Java程序时，你实际上做了什么工作？

- a. 将其存盘。
- b. 将其转换为计算机能够理解的格式。
- c. 将其加入到程序集合中。

2. 什么是变量？

- a. 是摇摆但是不下降的东西。
- b. 程序中被编译器忽略的文本。
- c. 程序中用来存储信息的地方。

3. 修复错误的过程叫什么？

- a. 解冻。
- b. 调试。
- c. 分解。

2.11.2 答案

1. b. 编译是将.java文件转换为一个或一组.class文件。
2. c. 变量是用来存储信息的地方，后面将介绍其他存储信息的地方，如数组和常量。钟摆会摇摆但不会掉下来；注释是编译程序中被编译器忽略的文本。
3. b. 由于在计算机程序中的错误叫bug，修复这些错误就叫调试（debugging）。有些编程工具自带了名为调试器（debugger）的工具，可帮助修复错误。NetBeans自带了一个调试器。

2.12 练习

如果想更多地探索本章介绍的主题，可完成下列练习。

- 将英文短语“Hello world!”翻译成其他语言，翻译时可以使用谷歌翻译，地址为<http://translate.google.com>。编写一个程序，让计算机用法语、意大利语或葡萄牙语向世界问候。
- 在Saluton程序中故意添加一两个错误。例如，删除行尾的分号或将某行中的println改成print1n。保存文件再编译程序，比较由此产生的错误消息。

要获悉这些练习的答案，请访问本书的配套网站

www.java24hours.com。

第3章 Java之旅

本章介绍如下内容：

- Java的历史；
- 使用Java语言的好处；
- 几个Java示例。

在进一步探索Java编程之前，有必要更详细地了解Java这门编程语言，以及当今的Java程序员都在做些什么。尽管Java已经不像最初那样局限于Web浏览器编程，但是我们仍然可以找出一些如何将Java用于Web的有趣示例。

本章将访问一些以Java程序为主的网站，并讨论该语言的历史和发展过程。

要开始这次Java之旅，必须有能够运行Java程序的Web浏览器。

启动你选择的浏览器，穿上舒适的衣服，准备开始Java之旅。因为不需要离开房间，所以你不会体验到观光旅游带来的那种简单的快乐，比如异国风情、美食珍馐等。

但是，这也有好的一面：没有兑换旅行支票的麻烦、不需要护照，而且也没有蒙特祖玛（注：一款游戏的名称）式的报复。

3.1 第一站：Oracle

Java之旅的第一站始于Oracle公司创建的www.java.com，Oracle公司管理着Java语言。

作为Web页面的一部分运行的Java程序称作applet。在网页中放置applet就像放置其他页面元素（比如照片和文本）一样：使用标记语言HTML指定在哪里显示程序、它多大以及程序在运行时做什么。Java还可以另外两种方式增强了Web页面：用Java语言编写的桌面程序可以从Web浏览器中启动；Web服务器运行的Java servlet可以分发Web应用程序。

图3.1显示的是一款使用Java语言编写的大型多人在线游戏RuneScape。通过使用任何Web浏览器来访问www.runescape.com网站，即可免费玩这款游戏。当访问该网站时，你可以在主页面上运行并玩这款游戏。该程序是使用Java语言实现的。

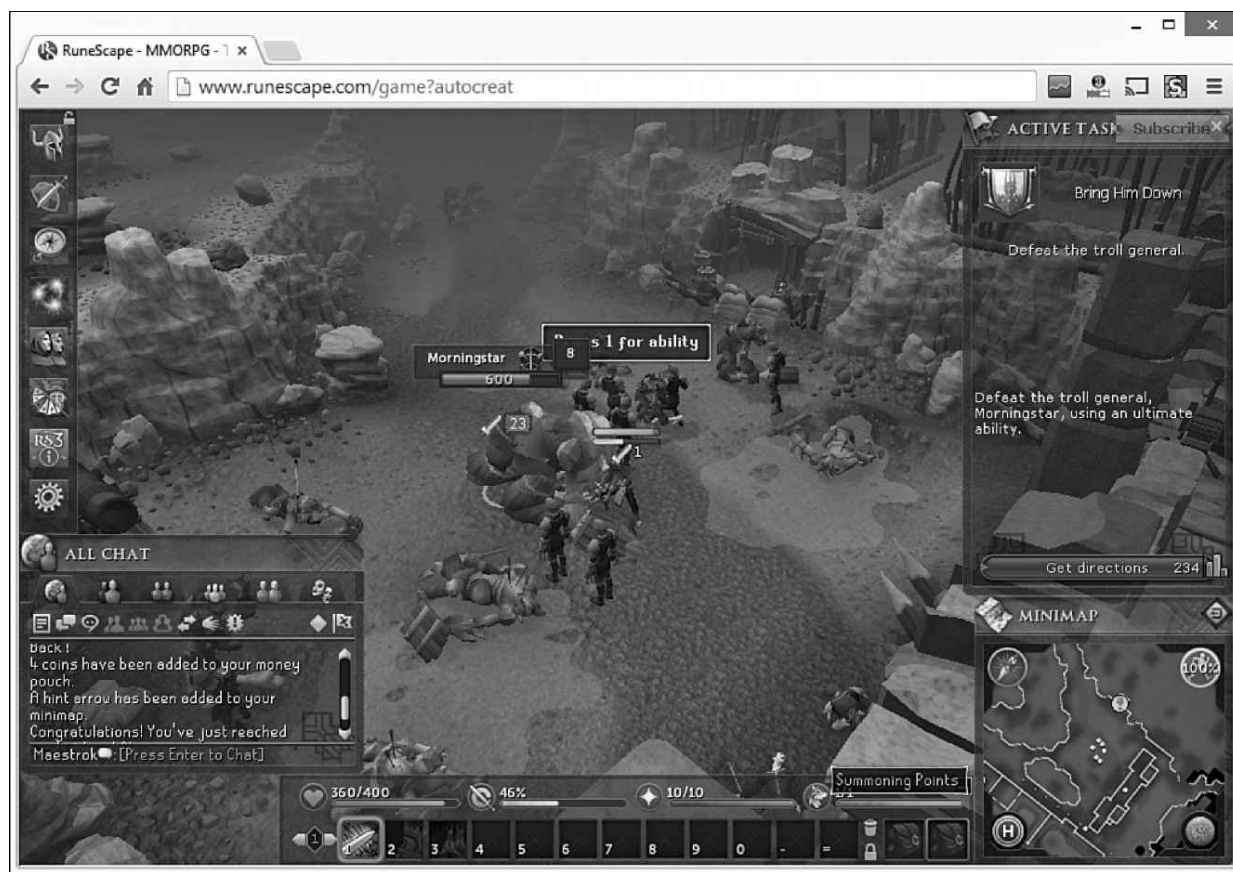


图3.1 由Java语言编写的在线游戏Rune Scape

当运行该游戏时，它将在几秒钟之内加载完毕，然后你就可以创建一个角色，探索一个幻想的世界了。

Oracle的Java部门主导着Java语言的开发。通过Oracle的Java.com网站，可以免费下载*Java Magazine* 在线杂志。该杂志展示了如何在Web站点、Android手机和其他平台上使用Java。现在有几十亿台设备运行的是使用Java编写的程序。

Java.com站点可以帮助人们安装能够运行applet的浏览器增强特性。该站点还描述了Java语言当前在Internet上的使用方式。

Oracle还为Java开发人员提供了一个更为技术性的Web站点，其网址为www.oracle.com/technetwork/java。读者可以在该站点上找到最新版本的NetBeans、Java Development Kit，以及其他编程资源。

Did you Know?

提示

Oracle在JDK和其他产品中包含了Java插件，因此该插件可能已经安装到你的计算机上。如果不确定是否安装了Java插件，可以访问www.java.com。点击“Do I Have Java”链接可以检测Java插件是否存在。如果没有找到，则该网站会提示你进行下载。

Java历史简要回顾

当Sun公司创造了Java语言时，当时的公司主管之一Bill Joy将该语言称为“15年努力的成果，它是更好、更可靠的计算机编程语言”。Java的创建过程比他说的还要复杂一些。

Java是在1990年由James Gosling创建的，其初衷是作为智能设备（如交互式电视、无所不能的烤箱、时间旅行的终结者、奴役人类的SkyNet军用卫星等）的大脑。Gosling对其使用C++编写的程序感到失望，他灵机一动，决定躲在办公室开发一种更适合其需求的新语言。

By the Way

注意

读者可能已经听说过Java是Just Another Vague Acronym（只是另外一种模糊的缩写）的首字母缩写，或许还听说过Java是以Gosling最喜欢的咖啡命名的。

其实，Java语言的命名没有任何秘密，也不是出于对咖啡这种饮品的热爱。之所以选择以Java来命名，其原因与喜剧演员Jerry Seinfeld喜欢说salsa这个单词一样：它听起来很酷。

Gosling最初将这种语言命名为Oak，灵感来自Gosling从办公室的窗户向外看到的一棵橡树。当时交互式电视已成为一个具有数百万美元产值的行业，该语言成为公司进入该行业的发展策略的一部分。可到今天这些都没成为现实（尽管Netflix、Hulu以及其他公司都在向游戏领域进军），但Gosling发明的新语言却发生了重大变化。正当Oak开发小组即将被放弃之时，Web开始流行起来。

正是阴差阳错，使Gosling发明的语言适合家用电器的特性也使其适用于Web。Gosling的团队设计了能够让程序在Web页面安全运行的方法，而且为了与该语言的新用途相匹配，为之选了一个引人注目的新名字：Java。

虽然Java还可以做很多其他的事情，但Web为Java提供了舞台，吸引了全世界开发人员的注意。当Java语言成为主流语言时，如果你没听过Java，那一定是离群索居或在从事长期的轨道任务。

Java语言总共有如下9个主要版本，每个版本都有一些重要的新特性。

- Java 1.0: 最初的Java版本（1995年）。

- Java 1.1: Java数据库连接（JDBC），图形用户界面得以提升（1997年）。
- Java 2 1.2版：内部类、用于Web浏览器的Java插件和数据结构（1998）。
- Java 2 1.3版：增强了多媒体功能（2000年）。
- Java 2 1.4版：增强了对Internet、XML处理和断言的支持（2002）。
- Java 5：泛型编程、新的循环、注释和自动数据转换（2005）。
- Java 6：内置了Derby数据库和Web服务（2006）。
- Java 7：改进了内存和资源管理，增加了Nimbus图形用户界面（2011）。

最新的版本——Java 8，于2014年3月发布。经过3年的改进，这一新版本引入了闭包，对Java高级编程来说，这是一个热切期待的特性，第20章将会讲到。而且它具有处理日期和时间的更好方式，以及添加了一些新的集合和注释。

如果你不知道这些东西是什么——比如内部类、泛型编程或XML——不要害怕，本书后面会讲到这些东西。

By the Way

注意

你可能好奇Java版本的编号为什么如此奇怪，比如直接从2跳跃到5，将第7版称为Java 6，而且在某些版本号还包含整数和小数。我对此也很好奇！

幸运的是，从2006年开始，Java的版本编号方案开始具有意义。每一个新版本发布时，其版本号都是整数，而且比上一版大1。

3.2 去Java学校

Web为教育工作者和学生提供了众多的资源。由于与标准Web页面相比，Java程序能够提供更多的交互体验，因此有些程序员使用Java编写用于Internet的学习程序。

读者可访问<http://www.cs.ubc.ca/~van/sssjava>来看这样的一个示例。通过该网站，可以访问由Michiel van de Panne（英属哥伦比亚大学计算机科学专业的教授）开发的跳台滑雪模拟器。该程序使用Java来演示当滑雪人员从不同的斜坡进行跳台滑雪时，其基于物理的动画。通过将鼠标向8个方向移动，可以控制滑雪人员的动作，而且这8个方向都会对滑雪的成功与否造成影响。图3.2显示的是在我的虚拟滑雪人员遇到阻挡之前，一次成功的滑雪尝试。

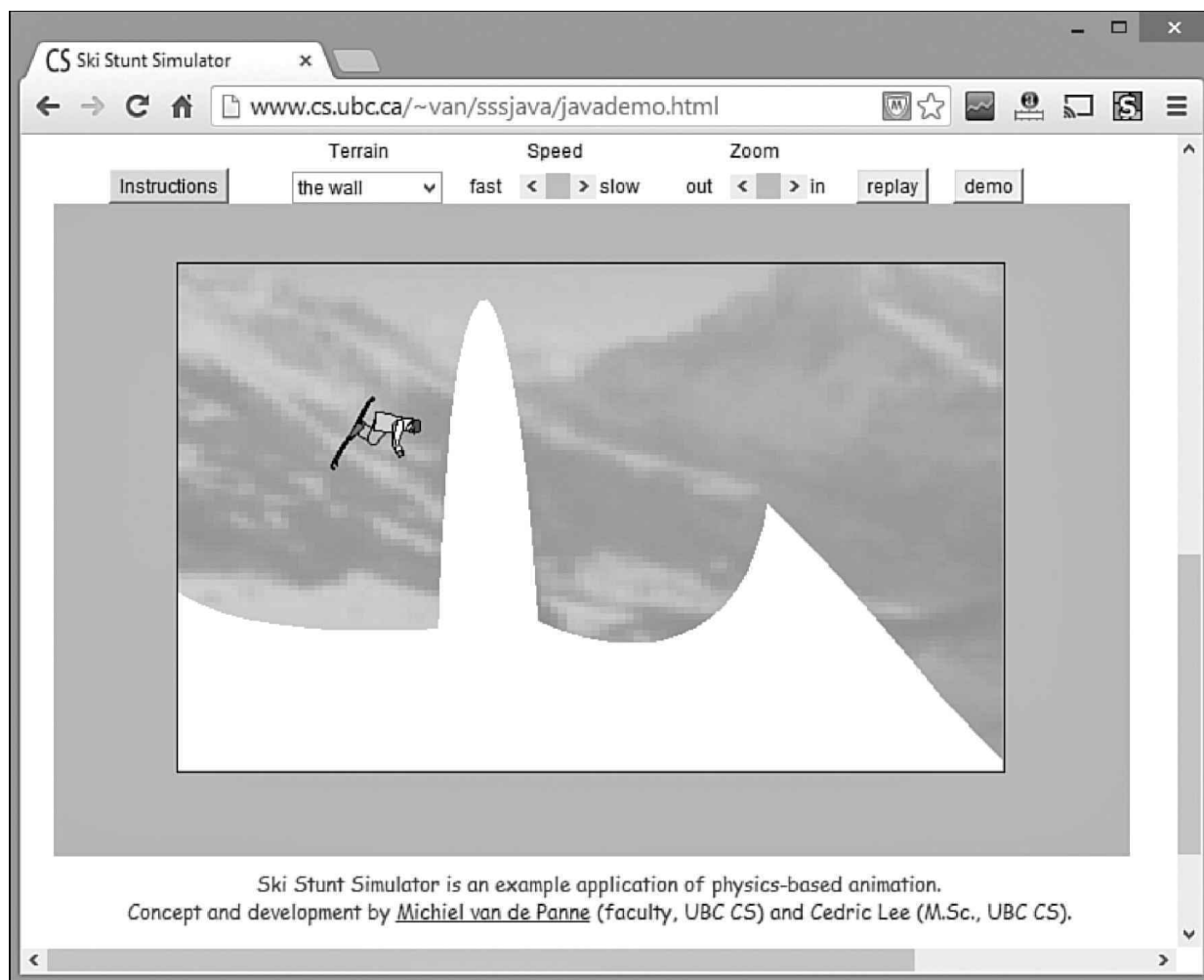


图3.2 使用Java语言编写的跳台滑雪模拟器可以提供交互体验

Watch Out!

警告：

当你访问该网站时，将运行applet，你可能会遇到“应用被阻止（Application Blocked）”安全警告。在本书付印前不久，Oracle修改了Java插件处理不包含数字签名（用于验证身份）的applet的方法。现在applet运行失败，并带有一个警告消息，而不是询问用户是否运行。

你可以调整Java安全设置，以允许运行特定Web页面的applet。

尽管有大量的教育程序可运行在不同的操作系统上，但使这种程序脱颖而出的是其可用性（availability）。模拟器是直接**在Web页面上**运行的，它和大多数桌面软件不同，不需要专门安装，也不局限于特定的操作系统。只要计算机中安装了**JVM**，那么就可以运行**Java**程序。

浏览器载入的**JVM**与第2章中运行**Saluton**程序使用的**JVM**相同。浏览器的**JVM**只能运行**Web**页面中的**Java**程序，而不能处理位于其他地方（比如台式机）的程序。

第一款支持**Java**的浏览器需要有内置的**JVM**。如今，浏览器是否支持**Java**则依赖于是否有**Java**插件。**Java**插件是一种作为浏览器的增强特性运行的**JVM**。

诸如跳台滑雪模拟器这样的**Java**程序不需要针对特定的操作系统进行编写，由于类似于**Windows**这样的操作系统也称为平台，因此**Java**的这种优势称之为平台独立性。**Java**可以在多种系统上运行。**Java**的开发者们之所以认为它必须支持多种平台，是因为要将其用于各种家用电器和其他电子设备。

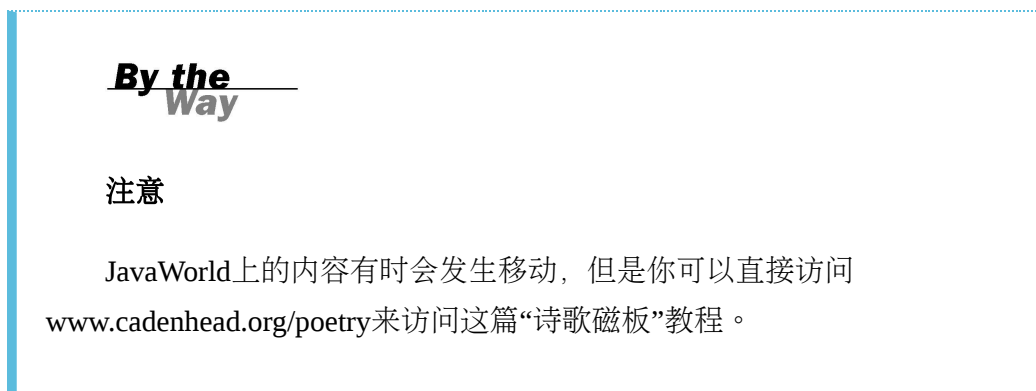
用户可将使用**Java**语言编写的程序运行在各种系统上，而且不需要做任何额外的工作。在正常情况下，使用**Java**编写程序时，不需要为不同的操作系统和设备创建不同的版本。

3.3 在JavaWorld用午餐

通过前面的介绍，读者对**Java**的兴趣应该逐渐浓厚起来。我们现在可在**JavaWorld**吃午餐。**JavaWorld**是一本针对**Java**程序员的在线杂志，

其网址为www.javaworld.com。

JavaWorld提供了“如何（how-to）”文章、新闻报道，以及与Java开发热点领域相关的研究中心。以Web格式出版的一个优点是，可以结合文章演示Java程序。图3.3所示为一个“诗歌磁板”Java程序，是在描写如何创建该程序的文章中提供的。



JavaWorld发表了与Java语言及其发展相关的文章和评论。自Java语言面世以来，一个激烈争论的问题是该语言是否安全。

当Java程序位于Web页面中时，由于其工作方式的原因，安全性至关重要。读者在本章尝试的Java程序将下载到你的计算机中。当程序下载完毕后，将在计算机上运行。

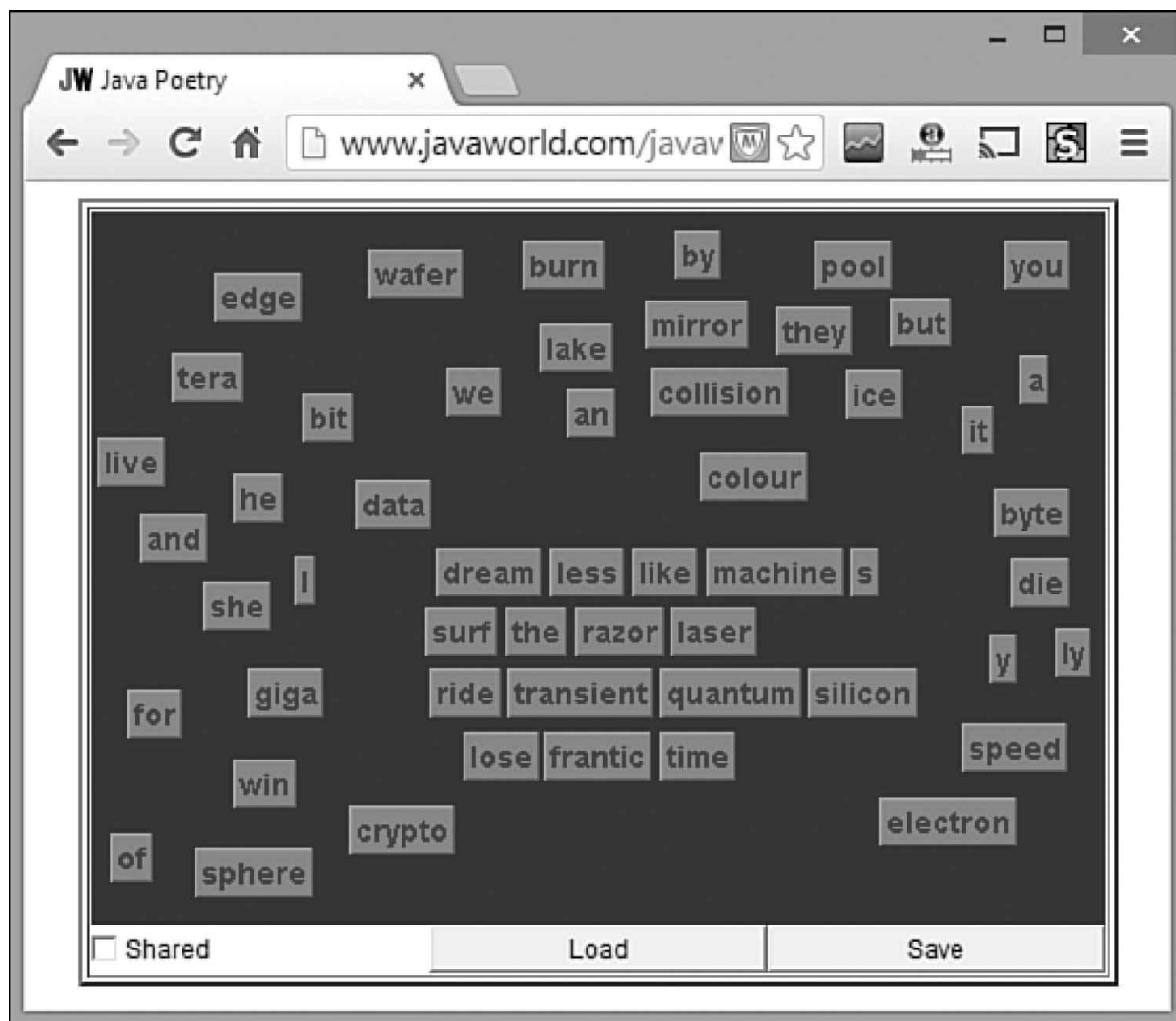


图3.3 JavaWorld中一篇介绍如何创建“诗歌磁板”的文章包含了这个程序示例

除非你认识很多人，否则你访问的大部分Web页面都是由陌生人发布的。从安全角度看，运行这些人的程序无异于将你的计算机借给别人使用。如果Java语言不能防范这种滥用行为，Java程序将把病毒引入到你的计算机系统，删除文件、播放William Shatner的歌曲，以及执行其他不可告人的事情。

Java包含了几种不同类型的安全措施，确保运行在Web页面中的Java程序是安全的。它的安全性是对通过Web运行的Java applet进行下

列限制来实现的：

- 任何applet都不能打开、读写或删除用户系统中的文件或系统属性；
- 任何applet都不能运行用户系统中的其他程序；
- applet创建的所有窗口都明确标识为Java窗口；
- 除其所属的网站，applet不能连接到其他网站；
- 所有applet都需要进行验证，确保编译后未被修改。

一般而言，Java语言已经被认为是足够安全的，可以在Web上使用，但是近几年发现的安全漏洞还是使得某些安全专家建议用户在其浏览器上完全关闭Java。为了解决这些问题，Oracle为Java升级工具提供了Java插件，以确保在发现漏洞时，能够进行升级，从而阻止漏洞。

Java还为在浏览器中运行的程序提供了更灵活的安全策略。可以将某些公司和程序员指定为“可信的开发者”，从而在你的浏览器中运行他们的Java applet，而不受正常的限制。

这种信任系统是通过使用具有数字签名的applet（数字签名是可以明确识别Java程序作者的文件）建立的。这些签名是与独立认证机构（如VeriSign）联合创建的。

如果读者曾经授权一个程序在浏览器（比如IE或Google Chrome）中运行，也就相当于建立了信任与身份验证的系统。

尽管applet在今天仍然有用武之地，但是这几年发展起来的其他技术，比如Flash、Silverlight、HTML5等已经在基于Web页面的程序中崭

露头角。Java在移动应用、服务器程序和桌面软件中的应用却越来越常见。

3.4 在NASA仰望天穹

Java之旅的第一天下午将前往NASA——一家大量使用Java的美国政府机构。其中最注明的一个例子是SkyWatch，这是一个帮助天文学家观察轨道卫星的applet。通过访问www.cadenhead.org/nasa，你将自动被转接到NASA的SkyWatch站点，然后即可将该applet载入到浏览器中。

SkyWatch将卫星（也可以自行添加或移除卫星的数量）的当前位置和轨道叠加到世界地图上。图3.4中运行的applet显示了星系演化探测器（GALEX）卫星在早上6点之前通过水瓶星座的运动轨迹（持续时间大约为3分钟）。

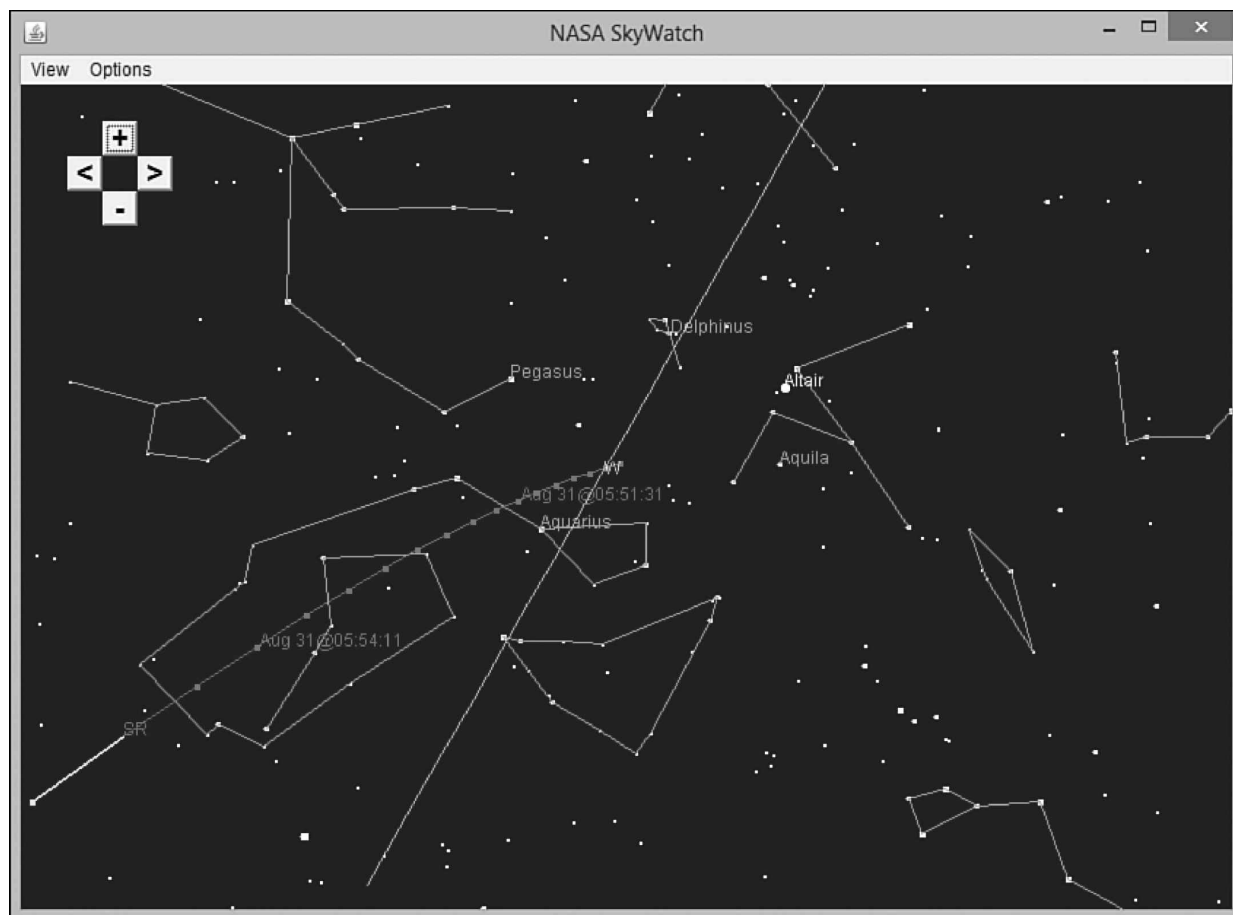


图3.4 NASA的SkyWatch applet可以监视轨道卫星的位置和路径，这对卫星观测者来说是一个福音

这个applet可以不断重绘出轨道卫星在运行时的位置。这种实时更新是可以实现的，因为Java语言是多线程的。多线程是计算机同时执行多项任务的一种方式；程序的一部分负责一项任务，另一部分负责另一项任务，两部分互不影响。在这种情况下，程序的每一部分称为一个线程。

在诸如SkyWatch这样的程序中，每一颗卫星都运行自己的线程。如果使用诸如Windows 8这样的操作系统，在同一时刻运行多个程序时，使用的就是这种行为。如果你在一个窗口中玩Desktop Tower

Defense游戏，同时在另外一个窗口中查看公司销售报表，同时还给朋友打长途电话——恭喜你，你是多线程的。

3.5 回归正题

此时，读者可能会有这样的印象：Java主要在空间爱好者、诗人和滑雪人员中使用。Java之旅的下一站将展示一个Java的常规使用示例。

将Web浏览器指向QuoteMedia站点，网址为www.quotemedia.com。这个公司提供的一个Java applet可以在其他网站上显示股票行情。图3.5所示为其滚动股票行情的一个演示版本（要亲自察看，可以访问www.cadenhead.org/stocks，然后会被重定向到这个页面）。

不像其他股票分析程序那样，需要在每个要访问它的雇员的计算机上安装软件。通过使用Java，QuoteMedia的雇员可以使用Web浏览器来访问这些程序。所有雇员所需要做的就是只需访问该公司的网站。

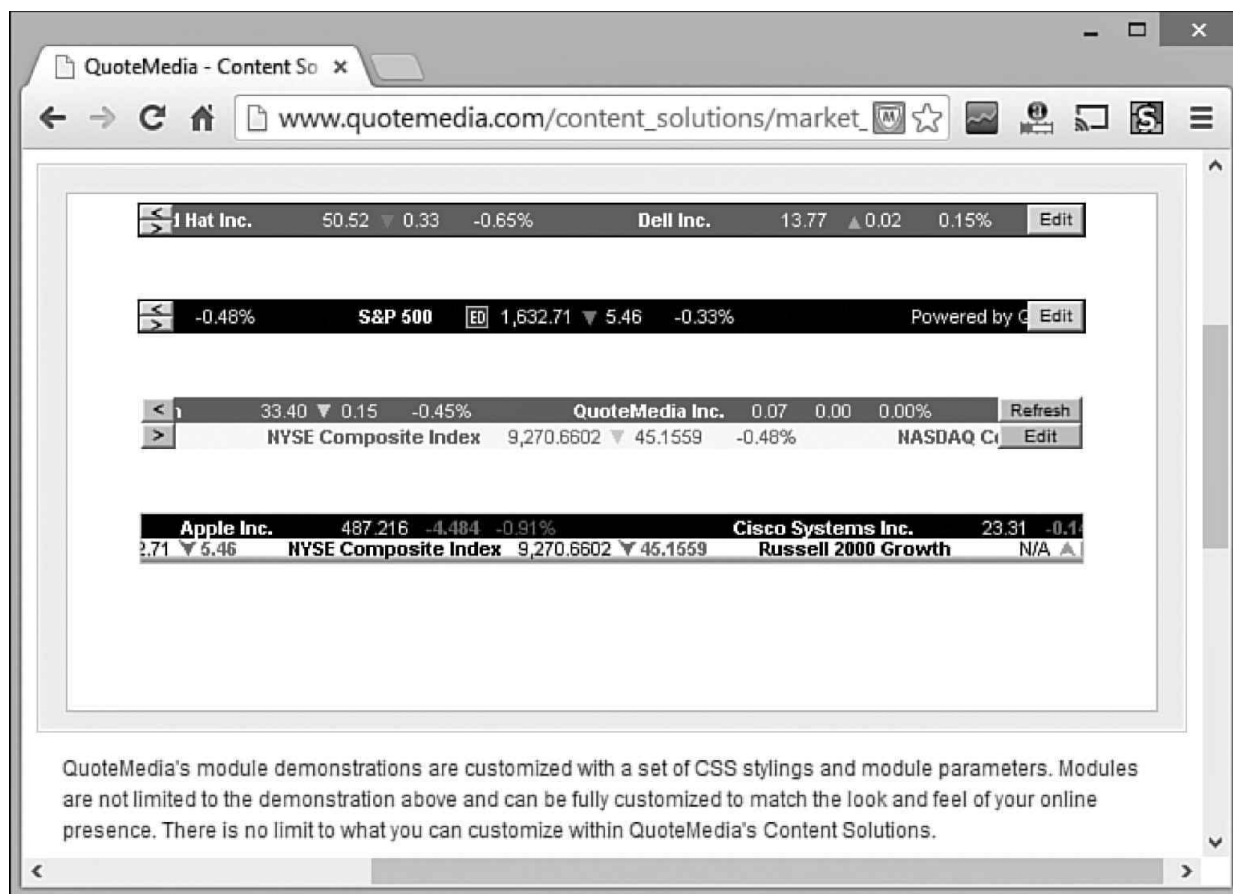


图3.5 QuoteMedia开发的显示股票行情的Java程序

可以通过不同的方式看待像这个applet的程序。一种方式是认为该程序就像物体——存在于现实中，占据一定的空间，并具备特定的功能。Java使用的面向对象编程（OOP，将在第10章介绍）将计算机程序作为一组对象来创建。每个对象处理特定的工作，并且知道如何同其他对象交流。例如，股票行情程序可以有如下对象组成：

- 一个报价对象（quote object），它表示一个单独的股票报价；
- 一个组合对象（portfolio object），它用来存储特定股票的一组报价；
- 一个股票对象（ticker object），它显示一个组合（portfolio）；

- 一个Internet对象、一个用户对象和众多其他的对象。

在这个模型下，股票行情软件是一个集合，这个集合包含完成工作所需的所有对象。

OOP是一种功能强大的程序创建方法，让编写的程序更有用。来看股票行情软件，如果程序员想将其报价功能加入到其他软件中，报价对象不需要做任何修改就可以用于新程序中。

使用对象编写的程序易于维护，因为其组织方式更好。对象包含了完成工作所需的数据和代码。对象也使得程序更具扩展性。可以仿照现有的对象生成一个新对象，然后为其添加新的功能。

3.6 到SourceForge去问路

这次Java程序的世界之旅一直都是由精通基于Web技术的难点和特点的专家引路，稍后读者将自己探索旅程，所以先停下来，了解一个对程序员来说非常有用的站点，这就是SourceForge网站，在这里可以找到使用Java（或其他任何编程语言）编写的复杂程序示例，其网址为www.sourceforge.net。

SourceForge是一个致力于协同编程项目的大型网站。如果你有兴趣与他人一起编写程序，你可以在SourceForge上开始一个项目，共享所有的文件，招募人手并与他们进行沟通。该网站上有300,000多个项目都是开源的，这意味着程序员需要共享所有的源代码。所谓的源代码，就是用来创建计算机程序的文本文件的另外一个名称。你在第2章开发的Saluton.java文件就是一个源代码的例子。

如果你使用SourceForge主页顶部的搜索框来搜索Java applet，你将在站点的项目目录中找到7000多个列表。

SourceForge上的一个程序是Aleksey Udovydchenko的Absolute游戏。这是一个具有街机风格的太空游戏，你控制着战舰炸出一条道路，然后穿过小行星带（见图3.6）。该游戏具有滚动的动画、图形、键盘控制和声音等特色。访问www.cadenhead.org/absolute可以进一步了解该游戏的详情。访问www.sourceforge.net/projects/absolutejava，可以下载该游戏的源代码。

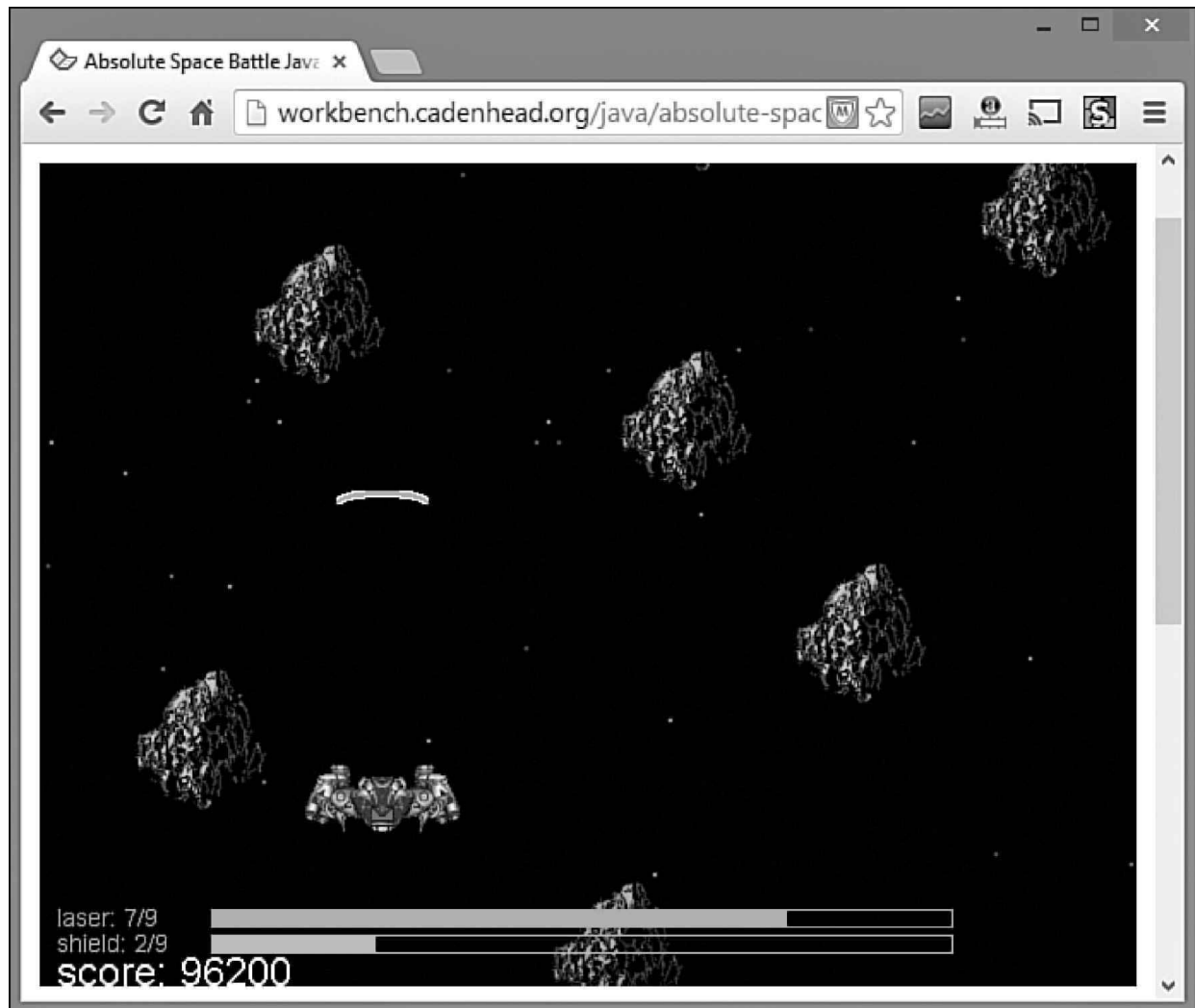


图3.6 Absolute这款Java程序的源代码可以在Source Forge中找到

整个Absolute程序的源代码只有700行出头。考虑到该程序的功能，代码行数已经非常小了。该程序能够运行，是因为Java包含了大量的类库，你可以在自己的程序中使用。Udovydchenko调用了Image类来显示小行星这样的图形，还调用了AudioClip类来播放激光枪发射和爆炸的声音。

这么多程序之所以使用Java开发（无论是SourceForge上的程序还是其他地方的），是因为该语言功能强大，而且容易学习。

Java语言的最初目标之一就是要比C++容易掌握，James Gosling于20世纪90年代在他的智能家电项目中使用的就是C++。Java的很大一部分都是基于C++的，因此学习过C++语言编程的人学习起Java来也不困难。然而，C++中有些难以学习和难以正确使用的内容已经从Java中删除。

对首次学习编程的人来说，Java要比C++易学。有些语言是为了让经验丰富的程序员能够在程序中充分利用计算机功能而开发的。这些语言比较简洁，而且包含编程老手很容易理解的其他特性。

Java并没有使用这些特性，而是使其成为尽可能简单的面向对象编程语言。创建Java语言的目的是易学、易调试和易于使用。Java还包括了大量增强的特性，从而使得能够与其他语言相抗衡。

3.7 在手机上运行Java

这次旋风式Java之旅的最后一站是Google Android手机。运行在Android上的每一个单独的程序都是使用Java开发的。这些应用程序扩展了手机的功能，并被称之为app（应用）。其中最流行的一个app是称之为“愤怒的小鸟”的游戏，如图3.7所示。



图3.7 愤怒的小鸟和其他Android app是使用Java语言开发的

如果你对这款游戏还不了解，或者想深入了解这款游戏，请访问www.angrybirds.com（最好别去了解这款游戏，否则它会占用你一天、一周，甚至是一个月的时间——这要取决于你多么讨厌游戏中的那些猪！）。

之所以将Android作为Java之旅的最后一站，是因为Android成为Java语言使用最为广阔的一个领域。在掌握了Java语言之后，你可以使用Android软件开发包（SDK）开发自己的app。Android SDK是一款可以在Windows、Mac OS和Linux上运行的免费编程套件。

如今大约有250,000个app可以用于Android手机和运行移动操作系统的其他设备。第24章将会讲解如何创建这些app。

3.8 总结

现在，Java之旅到此结束了，是时候收起你的行李，并准备开始真正的Java编程了。

在接下来的21章中，你将会掌握Java语言的各种基本组件，学习如何使用面向对象编程的方式来创建对象，以完成任务，还将学习如何设计图形用户界面等内容。

3.9 问与答

问：为什么Java applet不再流行了？

答：当Java语言于20世纪90年代中期被发明时，大多数人学习该语言的目的是编写applet。当时Java是创建可在Web浏览器上运行的交互式程序的唯一方式。多年以来，各种替代技术层出不穷。

Macromedia Flash、Microsoft Silverlight，以及新发布的HTML5 Web标准都提供了在Web页面上运行程序的方式。

由于applet在载入到浏览器中时较为费时，而且浏览器开发人员在支持新版本Java方面总是慢人一拍，applet逐渐式微。尽管后面又引入了Java插件来在浏览器中运行Java的当前版本，但是Java在那个时候已经远离了它的初衷，开始演变成为一种复杂的通用编程语言。

问：Java SE（Java Standard Edition）和Java EE（Java Enterprise Edition）之间的区别是什么？Java EE需要花钱么？

答：Java Enterprise Edition是Java Standard Edition的一个扩展，它包含了支持Enterprise JavaBeans、XML处理和servlet开发的包，其中servlet是运行在Web服务器上的Java程序。Java EE还包含一个应用服务器，这是一个复杂的环境，用来执行专门为企业和其他大型组织开发的具有大量计算需求的Java软件。Java EE开发工具可以从www.oracle.com/technetwork/java/javaee上免费下载。

3.10 测验

如果此刻你的大脑还没休息，请回答下面的问题以测试对本章内容的理解程度。

3.10.1 问题

1. 面向对象编程因何而得名？
 - a. 程序由一组协同工作的对象组成。
 - b. 由于难于掌握，人们经常反对（object）它。

- c. 它父母给起的名字。
2. 下面哪项不属于Java安全性？
- a. Web程序不能运行用户计算机上的程序。
 - b. 总是验证程序作者的身份。
 - c. Java窗口都标识为Java窗口。
3. 计算机或设备需要什么才能运行Java程序？
- a. Java编译器。
 - b. Java虚拟机。
 - c. 两者都需要。

3.10.2 答案

1. a. 它也被缩写为OOP。
2. b. 程序员可以使用数字签名和诸如VeriSign等身份验证公司，但这不是必需的。
3. b. Java虚拟机（JVM）是一个解释器，可以将Java字节码转换为计算机或设备可以运行的指令。创建Java程序时会用到编译器，但是运行时不需要。

3.11 练习

打开行李前，读者可以通过下面的练习进一步探索本章的主题：

- 通过SourceForge站点（www.sourceforge.net）了解使用Java语言开发了哪些纸牌游戏；
- 访问Oracle为Java用户开发的网站www.java.com，然后单击“Do I Have Java?”链接。然后根据指令来查看你的计算机是否安装了Java。如果有必要的话，可以下载并安装Java的最新版本。

要获悉每章末尾的练习答案，请访问本书的配套网站

www.java24hours.com。

第4章 理解Java程序是如何工作的

本章介绍如下内容：

- 学习应用程序是如何工作的；
- 构成一个应用程序；
- 向应用程序传递参数；
- 学习Java程序是如何组织的；
- 使用Java类库；
- 在应用程序中创建一个对象。

在Java编程中需要做出的一个重要决策是，程序应该在哪里运行。有些程序将在用户的计算机上运行，其他程序将作为Web页面的一部分运行。

在本地计算机上运行的Java程序被称为应用程序，在Web页面上运行的程序被称为applet，而在移动设备上运行的程序被称为app。

在本章，你将创建一个应用程序并在计算机上运行。

4.1 创建应用程序

你在第2章编写的Saluton程序就是一个Java程序。接下来，你需要创建另外一个应用程序，用来计算一个数的平方根，然后将该值显示出来。

在NetBeans中打开Java24项目，开始创建一个新应用程序。

1. 选择File->New File，打开New File Wizard。
2. 选择“Java”分类，然后选择“Empty Java File”文件类型，然后单击Next按钮。
3. 输入类名“Root”。
4. 输入包名com.java24hours。
5. 单击Finish按钮。

NetBeans将创建Root.java应用程序，并在源代码编辑器中打开一个空文件，以使用户开始工作。输入程序清单4.1所示的所有代码，注意不要输入行号以及行号后面的冒号。行号的目的是方便对程序的某些部分进行描述。输入完毕后，单击工具栏中的Save All按钮保存文件。

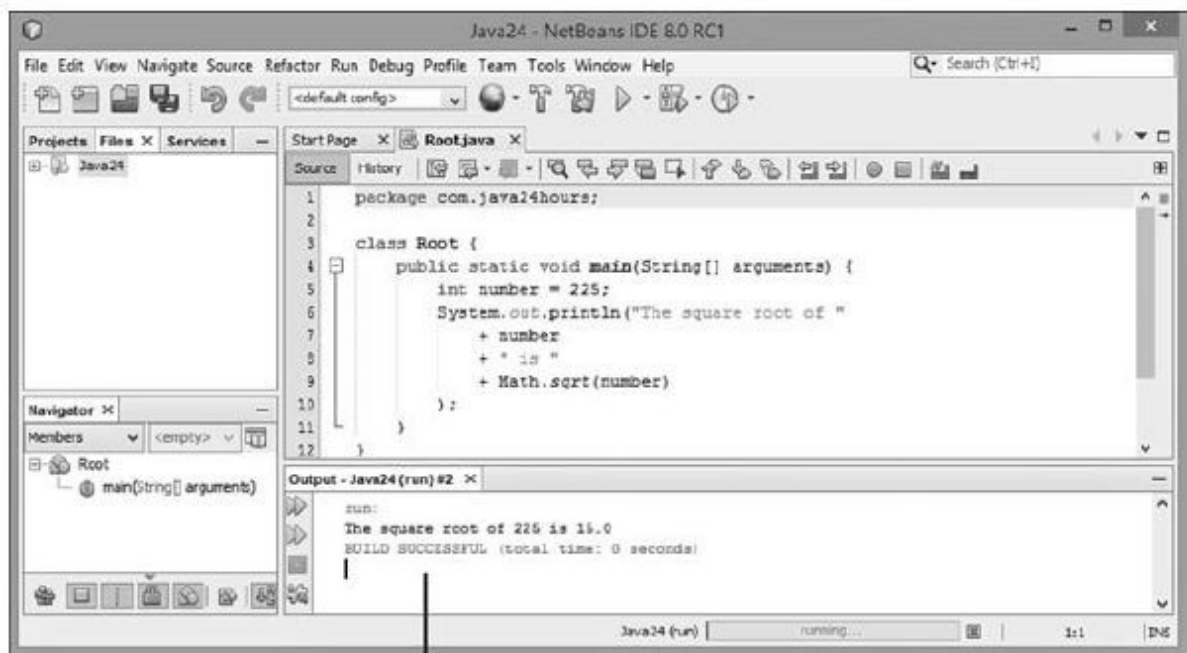
程序清单4.1 Root.java的完整版本

```
1: package com.java24hours;
2:
3: class Root {
4:     public static void main(String[] arguments) {
5:         int number = 225;
6:         System.out.println("The square root of "
7:             + number
8:             + " is "
9:             + Math.sqrt(number)
10:        );
11:    }
12: }
```

该Root引用程序完成如下任务。

- 第1行：应用程序位于com.java24hours包中。
- 第5行：在变量number中存储整数225。
- 第6~10行：显示该整数及其平方根。第9行的Math.sqrt(number)语句显示平方根。

如果你准确无误地输入了程序清单4.1中的代码（包含了所有标点符号，而且字母大小写没有问题），则可以在NetBeans中单击Run->Run File菜单命令来运行这个文件。该程序的输出结果将显示在输出面板中，如图4.1所示。



输出面板

图4.1 Root应用程序的输出

当运行Java程序时，Java虚拟机（JVM）将查找main()代码块，并开始处理该代码块中的Java语句。如果程序中没有main()代码块，则JVM将提示有错误。

第9行的Math.sqrt(number)语句演示了Java语言的一个内置功能——确定一个数的平方根。有一个名为Math的Java程序，它包含了名为sqrt()的方法，用来查找指定数值的平方根。

Math程序是Java类库的一部分，本章后面会讲到。

4.2 向应用程序传递参数

你可以使用java这个应用程序（会调用JVM）在命令行运行Java程序。当你在NetBeans中运行程序时，NetBeans会在幕后使用这个java程序。当以命令方式来运行Java程序时，JVM将装载该应用程序。而且命令可能会包含额外的信息，如下例所示。

```
java TextDisplayer readme.txt /p
```

发送给程序的额外信息称为参数。第一个参数（如果有的话）和应用程序名称之间用一个空格隔开，参数之间也用一个空格隔开。在上面的例子中，参数是readme.txt和/p。

如果想在参数内部包含一个空格，则必须将该参数使用引号括起来，如下例所示。


```
java TextDisplayer readme.txt /p "Page Title"
```

在该例中，TextDisplayer程序在运行时，具有3个参数：readme.txt、/p和“Page Title”。第3个参数中的引号可以防止Page和Title被视为两个独立的参数。

可以向Java应用程序传递任意数目的参数。然而，要使用这些参数，必须在应用程序中编写处理参数的语句。

为了解参数在应用程序中是如何工作的，在Java24项目中创建一个新类，步骤如下所示。

1. 选择File->New File。
2. 在New File Wizard对话框中，分别选择“Java”分类和“Empty Java File”文件类型。
3. 将该类命名为“BlankFiller”，将包命名为com.java24hours，然后单击Finish按钮。

在源代码编辑器中输入程序清单4.2中的代码，然后保存。编译该程序，纠正编辑器标记的任何错误（如果有）。

程序清单4.2 BlankFiller.java的完整代码

```
1: package com.java24hours;  
2:
```

```
3: class BlankFiller {
4:     public static void main(String[] arguments) {
5:         System.out.println("The " + arguments[0]
6:             + " " + arguments[1] + " fox "
7:             + "jumped over the "
8:             + arguments[2] + " dog."
9:     );
10: }
11: }
```

该程序编译成功，并可以运行。但是如果你使用菜单命令**Run->Run File**来运行该程序，将会得到一个看起来很复杂的错误，如下所示。

```
Output ▼
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 0
    at BlankFiller.main(BlankFiller.java:5)
```

之所以会发生该错误，是因为该程序在运行时，期待接收到3个参数。你可以在**NetBeans**中通过对项目进行自定义，来指定参数。

1. 选择菜单命令**Run->Set Project Configuration->Customize**，打开**Project Properties**对话框。

2. 在**Main Class**文本框中输入**com.java24hours.BlankFiller**。

3. 在**Arguments**字段中，输入“retromingent purple lactose-intolerant”，然后单击**OK**按钮。

由于你自定义了项目，因此在运行该程序时会有所不同。选择菜单命令**Run->Run Main Project**。应用程序将使用你指定的参数作为形容词来填充一句话，如图4.2所示。

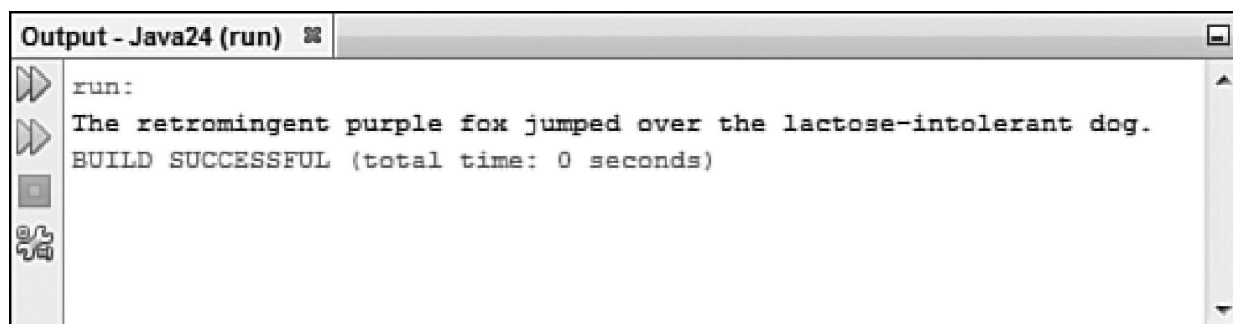


图4.2 BlankFiller应用程序的输出

返回**Project Properties**对话框，然后将自己选择的3个形容词指定为参数，而且要确保至少包含3个参数。

参数是一种自定义程序行为的简单方法。参数存储在称为数组的变量中。第9章将会讲到数组。

4.3 Java类库

本书解释了如何使用Java语言从头创建自己的程序。你学习了构成Java语言的所有关键字和操作符，并使用它们编写语句，从而让计算机做有趣和有用的事情。

尽管这是学习Java的最好方法，但是这有点类似于要构建一辆汽车，你需要先从头构建汽车的每一个零部件。

作为一名Java程序员，大量的工作已经为你完成了，前提是你知道去哪里寻找。

Java带有称之为Java类库的大量代码集合，你可以在自己的程序中使用它们。这个库是一个包含几千个类的集合，其中许多可以应用在你编写的程序中。

类可以采用类似于使用变量的方式在你的程序中发挥作用。

类用于创建对象，对象与变量相似，但是更为复杂。对象可以存放数据（这与变量一样），也可以执行任务（这与程序一样）。

Oracle在网上为Java类库提供了全面的文档，地址为<http://download.java.net/jdk8/docs/api>。该页面如图4.3所示。

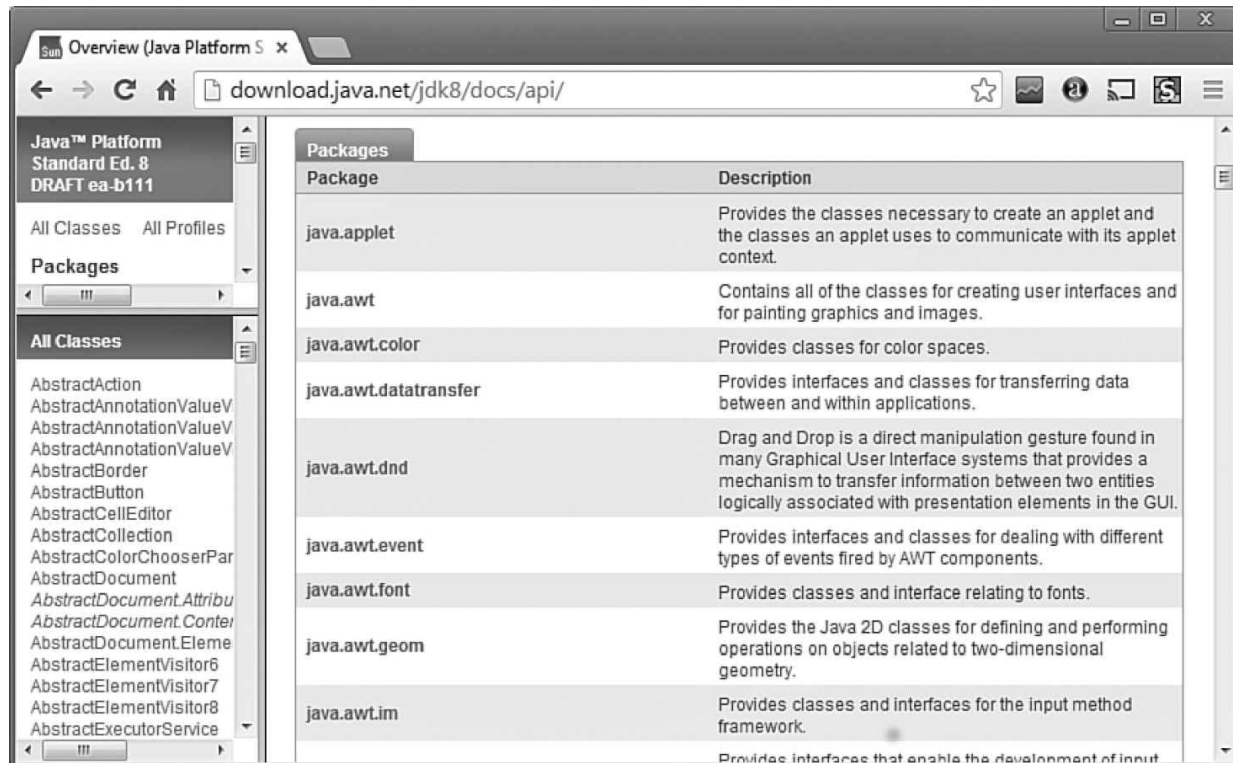


图4.3 Java类库的文档



Java类被组织成包，包的功能类似于计算机上的文件夹。目前为止你创建的程序都属于com.java24hours包。

Java类库文档的主页面划分为框架。最大的框架列出了组成Java类库的所有包以及相应的描述。

包的名字有助于描述其用途。例如，java.io是一组用于磁盘驱动器、Internet服务器和其他数据源的输入、输出的类；java.time包含与时间和日期相关的类；java.util包含了一些有用的实用工具类。

在文档主页面上，最大的框架中列出了一组包以及相对应的简短描述。单击包的名称可以获悉更多内容。页面中会载入包中包含的类。

在这个站点上，Java类库中的每一个类都有自己的文档页面。这个站点包含26000多个页面（你现在或永远都没有必要阅读所有的页面）。

在本章最后一个项目中，你可以随意查看类库并使用一个现有的Java类来做些事情。

Dice程序使用了java.util包中的Random类，它可以用来创建随机数。

要在程序中使用这个类，你要做的第一件事情是使用下面的语句导入包含这个类的包：

```
import java.util.*;
```

这样就可以在无需使用其全名（即java.util.Random）的情况下引用Random类。相反，你可以简单地以Random来引用它。上述语句中星号的作用是可以采用较短的名字引用包中的所有类。

Random类是一个模板，用来创建Random对象。要创建一个对象，需要使用new关键字，后跟类的名字和括号：

```
Random generator = new Random ();
```

这将创建一个名为generator的变量，它可以存放一个新的Random对象。这个对象是一个随机数生成器，可以产生一个或多个随机数。

在该程序中，将会使用对象的nextInt ()方法来产生一个随机整数：

```
int value = generator.nextInt ();
```

在Java中，整数范围为-2,147,483,648~2,147,483,647。生成器从这个范围中随机选择一个数，然后将其指派给value变量。

没有Java类库中的Random类，你不得不自行创建程序来产生随机数，而这是一个相当复杂的任务。随机数在游戏、教育程序和必须随机做些事情的其他程序中很有用。

在NetBeans中创建一个新的Java空文件，将其命名为Dice，然后放入到com.java24hours包中。当打开源代码编辑器后，输入程序清单4.3中的所有内容，然后单击Save按钮（或者选择菜单命令File->Save）。

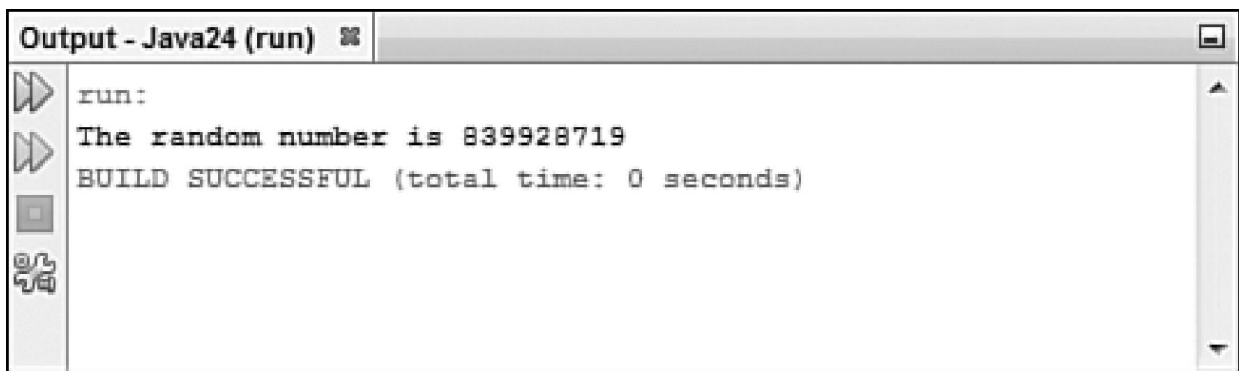
程序清单4.3 Dice.java的完整版本

```
1: package com.java24hours;
2:
3: import java.util.*;
4:
5: class Dice {
6:     public static void main(String[] arguments) {
7:         Random generator = new Random();
8:         int value = generator.nextInt();
9:         System.out.println("The random number is "
10:            + value);
11:     }
12: }
```

选择Run->Run File来运行该程序，输出如图4.4所示，当然，你看到的数值可能会与这里不同（实际上，出现相同数字的概率为40亿分之一，这要比彩票的中奖概率还低）。

与Java类库中的所有类一样，Random类在Oracle的网站上有可供阅读的大量文档。它描述了类的用途、所属的包、如何创建该类的对象，以及它有哪些方法可以用来做些事情。

可按照如下步骤查看该文档。



```
run:
The random number is 839928719
BUILD SUCCESSFUL (total time: 0 seconds)
```

图4.4 Dice程序的输出

1. 在Web浏览器中，载入页面
<http://download.java.net/jdk8/docs/api>。
2. 在主框架中向下滚动，直到看到java.util包的链接。单击该链接，将显示这个包的文档。
3. 在主框架中向下滚动，然后单击Random链接。

作为一名只有不到4小时编程经验的Java程序员，你可能会发现即使绞尽脑汁也难以理解这个文档。不要惊慌！这是写给有经验的程序

员看的。

在你阅读本书时，你可能会好奇Java的内置类是如何使用的，通过查看相关类的官方文档，你可能会获得一些有价值的信息。使用类的一个方法是查询类中包含的方法，每一个方法都执行一个作业。

在Random文档中，你可以滚动到nextInt()方法的解释。这个方法用在了程序清单4.3中的第8行。图4.5所示为页面上的相关内容。

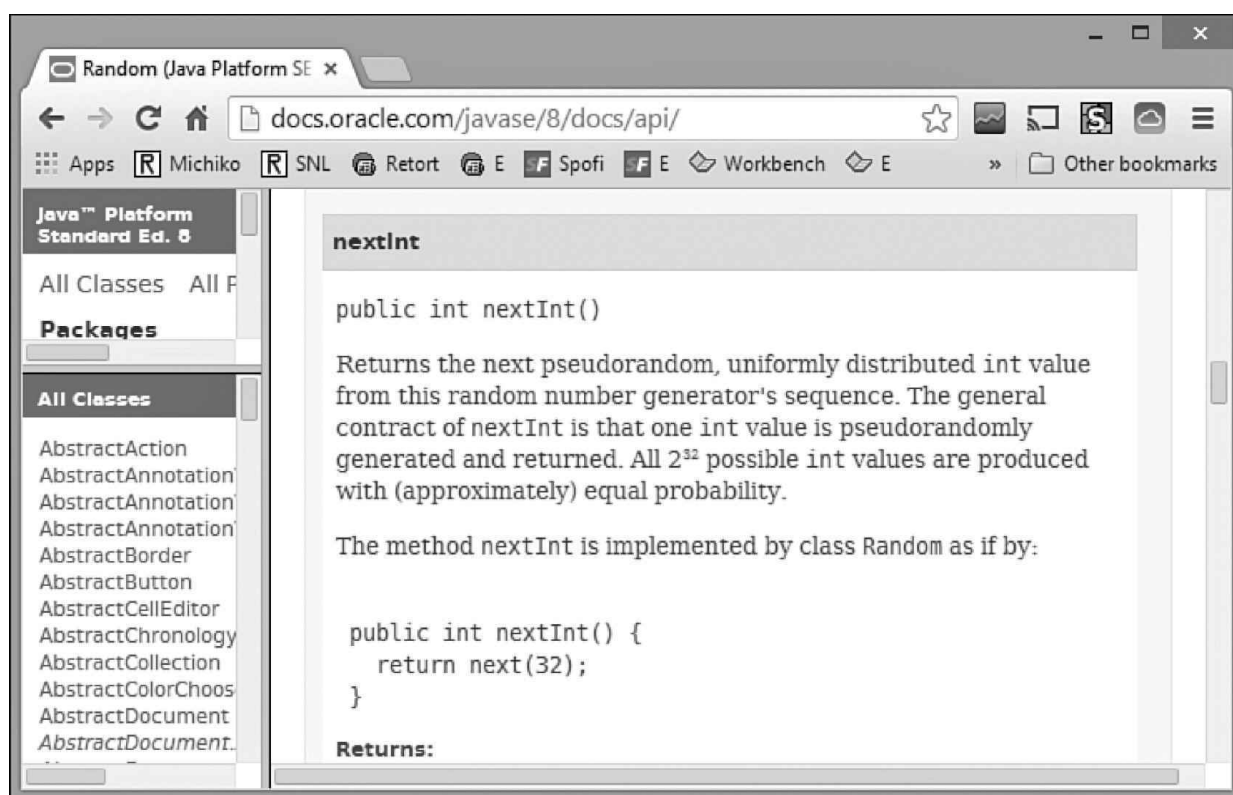


图4.5 Oracle针对Random类的文档

***By the
Way***

注意

本书中用到的所有Java类都在这个页面内进行了描述，在本书的学习中不需要这个在线文档也能成为一名Java程序员。但是本书用到的类还有一些其他特性超出了本书的范围，因此Java类库的文档可以辅助你的学习。

4.4 总结

在本章中，读者创建了一个Java应用程序，并向一个程序发送参数，还使用Java类库中的现有程序。

接下来的几章将继续介绍应用程序，让读者成为经验更丰富的Java程序员。应用程序的测试比较迅速，因为它们不需要你做任何额外的工作来运行它们（而在其他类型的程序中，你需要这么做）。

本章首次讨论了如何在Java程序中使用对象。你将在第10章返回该主题。

4.5 问与答

问：发送给Java应用程序的所有参数都必须是字符串吗？

答：应用程序在运行时，Java将所有参数存储为字符串。要使用整型或其他非字符串参数，必须将其进行转换，这将在第11章介绍。

问：既然applet是在Web页面中运行，应用程序可以在任何地方运行，那么Java Web Start启动的Java程序是什么呢？

答：Java Web Start是一种从Web浏览器启动Java应用程序的方法。用户通过单击Web页面上的一个链接来运行程序。与先下载，后运行安

装向导，然后再启动的桌面程序相比，这种方式无疑更为简便。

尽管是从浏览器中运行的，但是Java Web Start程序仍然是应用程序，而不是applet。应用程序永远都是最新的，因为它在每次运行时，都是通过Web从程序提供者那里下载到的。

Google Web Toolkit（GWT）是一组用于Web编程的开源工具，它可以将Java程序转换为JavaScript，从而使得它在Web浏览器中运行时速度更快、更可靠，而且还不需要JVM的介入。

4.6 测验

请回答下面的问题，以测试对本章内容的掌握程度。

4.6.1 问题

1. 哪种类型的Java程序可以由浏览器来运行？
 - a. applet。
 - b. 应用程序。
 - c. 两者都不可以。
2. JVM表示什么意思？
 - a. Journal of Vacation Marketing。
 - b. Jacksonville Veterans Memorial。

c. Java Virtual Machine 。

3. 如果你在给Java应用程序传递信息的方式上与他人发生争议，那么争议的焦点是什么？

a. 关于字符串的争论 。

b. 关于参数的争论 。

c. 关于功能的争论 。

4.6.2 答案

1. a. applet作为Web页面的一部分运行，而应用程序可以在任何地方运行 。

2. a、b或c。这是一个脑筋急转弯。这3个选项都可以用JVM来表示。但是Java Virtual Machine则是读者在接下来的20章中需要牢记的。

3. b. 应用程序以参数的方式接收信息。难道我们不能好好相处么？

4.7 练习

要应用应用程序的知识，建议完成下面的练习 。

- 根据Root应用程序，编写一个可以显示625的平方根的NewRoot应用程序 。

- 根据Root应用程序，编写一个NewRoot应用程序，它可以显示作为参数提交的那个值的平方根。

有关完成这两个练习的Java程序，请访问本书的配套网站
www.java24hours.com。

第5章 在程序中存储和修改信息

本章介绍如下内容：

- 创建变量；
- 使用不同类型的变量；
- 在变量中存储值；
- 在数学表达式中使用变量；
- 把一个变量的值赋给另一个变量；
- 递增或递减变量的值。

在第2章中，我们使用到了一个变量，它是一个用于存储信息的特殊位置。程序运行时，变量中存储的信息可以发生改变。在你的第一个程序中，变量中存储了一个字符串。字符串只是可在变量中存储的一种信息类型，变量还可以存储字符、整数、浮点数和对象。

本章将更详细地介绍如何在Java程序中使用变量。

5.1 语句和表达式

计算机程序是一组告诉计算机做什么的指令，每一个指令称为语句。下面就是Java程序中的一条语句：

```
int highScore = 450000;
```

在Java程序中，可以使用括号将一组语句编组，这一组语句称为块语句（**block statement**）。请看下面的程序片段：

```
1: public static void main(String[] arguments) {  
2:     int a = 3;  
3:     int b = 4;  
4:     int c = 8 * 5;  
5: }
```

其中第2～4行是一个块语句。第1行的左大括号表示块语句开始，第5行的右大括号表示块语句结束。

有些语句被称为表达式，原因是它包含一个数学表达式，并能得到一个结果。在上面的例子中，第4行就是一个表达式，因为它将变量c的值设置为8和5的乘积。在接下来的几节中，读者将会用到表达式。

5.2 指定变量类型

计算机在运行程序时，变量是计算机记住信息的主要方法。第2章的Saluton程序使用greeting变量来存储“Saluton mondo!”消息。计算机需要记住这段文本，以便稍后显示。

在Java程序中，创建变量所使用的语句必须包含下列两项内容：

- 变量名；

- 变量存储的信息类型。

变量也可以包含将要存储的信息的值。

要看不同类型的变量以及如何创建它们，可以运行**NetBeans**，然后创建一个新的空**Java**文件，其类名为**Variable**。

开始编写程序，输入下面的代码：

```
package com.java24hours;
class Variable {
    public static void main(String[] arguments) {
        // Coming soon: variables
    }
}
```

对其进行修改前保存该文件。

5.2.1 整数和浮点数

至此，程序**Variable**有了一个**main()**块，其中只包含一条语句：注释**//Coming soon: variables**。删除注释并输入下列语句：

```
int tops;
```

这条语句创建一个名为**tops**的变量，但没有给它赋值，因此该变量此时是一个空的存储空间。语句开头的**int**指定**tops**是一个用于存储整数

的变量。可以使用`int`类型来存储计算机程序所需的大多数整数，它能够存储 $-2.14 \times 10^9 \sim 2.14 \times 10^9$ 的任何整数。

在语句`int tops;`的末尾换行，然后输入下面的语句：

```
float gradePointAverage;
```

这条语句创建一个名为`gradePointAverage`的变量，其中`float`表示浮点数。浮点数变量用来存储可能包含小数的数值。

`float`变量类型可以存储高达38位的十进制数，而`double`变量类型可以存储高达300位的十进制数。

By the Way

注意

可以使用浮点变量来存储像2.25这样的绩点成绩（GPA）（我进入北德克萨斯大学的GPA就是2.25），还可用于存储像0这样的数字（我当年凭GPA进入一所好研究院的机会也是0）。不过，我在1996年找工作时，仍然被Sams出版社慧眼识珠，招募为他们的计算机图书作者。

5.2.2 字符和字符串

到目前为止，你所处理的变量都是数值型的，因此可能认为所有变量都是用来存储数字的。也可以使用变量来存储文本，在变量中可

以存储两种类型的文本：字符和字符串。字符是单个字母、数字、标点符号或其他符号。字符串是一组字符。

创建Variable程序的下一步是创建一个char变量和一个String变量。在语句float gradePoint Average; 后面加入下面两条语句：

```
char key = 'C';  
String productName = "Larvets";
```

在程序中使用字符值时，必须用单引号将赋给变量的字符值括起来，而对于字符串值必须用双引号括起来。

引号可以将字符或字符串同变量名或其他语句部分区分开来。请看下面的语句：

```
String productName = Larvets;
```

这条语句看起来好像告诉计算机，创建一个名为productName的字符串变量，并将文本值Larvets赋给该变量。但是由于没有使用双引号将单词 Larvets 括起，计算机认为要将productName的值设置为变量Larvets的值（如果没有Larvets变量，则程序将编译失败）。

加入char和String语句后，程序将如程序清单5.1所示。在更改之后，不要忘记存盘。

程序清单5.1 Variable程序

```
1: package com.java24hours;
2:
3: class Variable {
4:     public static void main(String[] arguments) {
5:         int tops;
6:         float gradePointAverage;
7:         char key = 'C';
8:         String productName = "Larvets";
9:     }
10: }
```

Variable程序中的最后两个变量在创建时，使用等号“=”赋予了初始值。在Java程序中，可以用这种方法给任何变量赋值，这将在本章后面看到。

该程序可以运行，但是不会产生输出。

By the ***Way***

注意

尽管其他变量类型都是小写字母（int、float和char），但创建字符串变量时，单词String的首字母必须大写。在Java程序中，字符串与变量语句中使用的其他信息类型不同，这将在第6章介绍。

5.2.3 其他数值类型的变量

目前为止，所介绍的变量类型是在Java程序中使用的主要变量类型。在特殊情况下，还可使用其他几种变量。

还有3种其他类型的整型变量。第一种为byte，取值范围为-128～127的整数。下面的语句是创建一个名为escapeKey的变量，其初始值为27：

```
byte escapeKey = 27;
```

第二种为short，可用于存储比int类型小的整数，其取值范围为-32768～32767的整数，如下例所示：

```
short roomNumber = 222;
```

最后一种数值变量类型的是long，通常用于存储int类型无法存储的整数。long变量可以存储 $-9.22 \times 10^{18} \sim 9.22 \times 10^{18}$ 的整数。

在Java中处理很大的数值时，很难一目了然地看到具体的数值，如下面的语句所示：

```
long salary = 2644000000;
```

除非你数一下0的个数，否则你很难看出它表示的是264.4百万美元。Java通过在数值中间使用下划线（_）来处理较大的数值，如下所示：

```
long salary = 264_400_000;
```

下划线将会忽略掉，因此变量的值没有发生变化，这只是一种让数值更容易阅读的方式。

Watch **Out!**

警告：

如果NetBeans IDE被设置为使用较老的Java版本，当在数值中使用下划线时会在源代码编辑器中被标记为错误。为了避免这种情况，可以在Projects（项目）面板中右键单击当前项目的名字（比如Java24），然后选择Properties（属性）。这将打开Project Properties对话框，而且在Categories面板中选中了Source。检查Source/Binary Format下拉列表，确保是Java的最新版本。

5.2.4 布尔变量

Java有一种称为布尔变量的变量类型，它只能存储true或false。乍一看，布尔变量好像不是特别有用，除非要编写大量的真假测验。然而，编写程序时，很多情况下都需要使用布尔变量。下面是一些可使用布尔变量解决的问题。

- 用户是否按下了某个键？
- 游戏结束了吗？
- 银行账户透支了吗？
- 这些裤子让我看起来更胖了么？
- 兔子能吃Trix吗？

布尔变量是存放回答为yes/no或问题为true/false的地方。

下面这条语句创建了一个名为gameOver的布尔变量：

```
boolean gameOver = false;
```

这个变量的初始值为false，这样的语句可在游戏程序中指示游戏还没有结束。然后，当导致游戏结束的事件发生时，将变量gameOver设置为true。

虽然在程序中，布尔变量的两个可能取值（true或false）看起来像字符串，但不应将它们用双引号括起。第7章将更详细地介绍布尔变量。

By the Way

注意

布尔数字是用George Boole（1815～1864）的名字命名的。Boole是一名数学家，在成年前靠自学成才。他发明了布尔代数，这是计算机编程、

数字电路和逻辑学的重要组成部分。可以想象，他在孩童时就能够做很好的真假测试。

5.3 给变量命名

在Java中，变量名可以以字母、下划线字符（_）、美元符号（\$）打头。名称的其余部分可以是任何字母或数字。你可以取任何喜欢的变量名，但应该采用统一变量命名规则。本节将介绍常用的变量命名方法。

在变量名的问题上，Java是区分大小写的，所以变量名在整个程序中必须大小写一致。例如，如果变量gameOver在程序的其他地方写成GameOver，编译程序时使用GameOver的地方将导致错误。

变量名应该描述变量的用途，第一个字母应该小写，如果变量名有多个单词组成，则将其他单词的首字母大写。例如，如果要创建一个整型变量，用于存储游戏中创记录的最高分，可使用下面的语句：

```
int allTimeHighScore;
```

在变量名中不能使用标点符号和空格，因此下面的变量名都是无效的：

```
int all-TimeHigh Score;  
int all Time High Score;
```

如果在程序中使用这些变量名，NetBeans将在源代码编辑器中的相应行处用红色警告图标来显示，表示这是一个错误。

By the Way

注意

变量名并不是Java区分大小写的唯一的一个地方。在程序中但凡你可以命名的东西，比如类、包和方法，大小写都必须一致。

5.4 在变量中存储信息

在Java程序中，可以在创建变量时就给它赋值，也可以以后再给变量赋值。

要在创建变量时就给它赋初值，可以使用等号（=）。下面的例子创建双精度浮点变量pi，且其初始值为3.14：

```
double pi = 3.14;
```

所有数值型变量都可以用类似的方法赋值。如果创建的是字符或字符串变量，必须像前面介绍的那样用引号将赋给变量的值括起来。

如果两个变量的类型相同，也可以将一个变量的值赋给另一个变量，请看下面的例子：

```
int mileage = 300;  
int totalMileage = mileage;
```

首先，创建了一个整型变量`mileage`，并且其初始值为300。接下来，创建了另一个整型变量`totalMileage`，并将变量`mileage`的值赋给它。这两个变量的初始值相同，都是300。在接下来的几章中，将介绍如何将一个变量的值转换为另一个变量的类型。

Watch ***Out!***

警告：

如果没有给变量赋初始值，则在另外一条语句中使用它之前，必须先赋值。如果没有赋值，则在程序编译时将报错，并显示如下错误消息“变量可能还没有初始化”。

前面已经学到，Java有多个相似的数值变量，来存储不同大小的数值。int和long都存储整数，但是long存储的数值范围更大。float和double都存储浮点数值，但是double类型存储的范围更大。

你可以在数值后面追加一个字母来指示数值的类型，如下面的语句所示：

```
float pi = 3.14F;
```

数值3.14后面的F表示这是一个float浮点数值。如果忽略了这个字母，Java会认为3.14是一个双精度浮点数值。

字母L表示长精度整数，D表示双精度浮点数值。

Java的另外一种命名约定是，值不变的变量名全部大写。这些变量被称为常量，下面创建了3个常量。

```
final int TOUCHDOWN = 6;  
final int FIELDGOAL = 3;  
final int PAT = 1;
```

由于常量的值不变，读者可能会问，为何使用常量，只需使用赋给常量的值即可。使用常量的优点之一是，程序更容易理解。

在上面的3个语句中，常量的名字是大写的。尽管这在Java中不是必需的，但是编程人员已经将其作为区分常量和变量的一种标准约定。

5.5 运算符

通过使用+、-、*、/和%等运算符，就可以在语句中使用数学表达式。在Java程序中使用这些运算符，就可以处理数值运算。

在Java中使用+运算符的加法表达式语句如下所示：

```
double weight = 205;  
weight = weight + 10;
```

上面第2条语句使用+运算符将weight变量的值设置为weight变量的初始值与10的和。使用-运算符的减法表达式如下：

```
weight = weight - 15;
```

该表达式将weight变量的值设置为weight变量的初始值减去15后的结果。

使用/运算符的除法表达式如下：

```
weight = weight / 3;
```

该表达式将weight变量的值设置为其初始值除以3后的结果（取整）。

要从除法表达式中找到余数，可以使用%运算符（也称为取模运算符）。下面两条语句可以求得245除以3之后的余数。

```
int remainder = 245 % 3;
```

乘法表达式使用的是*运算符。下面这条复杂的语句使用了乘法表达式。

```
int total = 500 + (score * 12);
```

表达式`score * 12`中，变量`score`与12相乘。这条完整的语句是将变量`score`与12相乘，然后再加500，最后将结果赋给`total`变量。如果`score`的值为20，则`total`变量的最终值是740，也即 $500 + (20 * 12)$ 。

5.5.1 变量的递增与递减

在程序中，常见的任务是将变量的值加1或减1。其中加1的操作称为变量递增，减1的操作称为变量递减。这些任务可以使用相应的运算符来完成。

使用++运算符可以将变量加1，如下面的语句所示：

```
power++;
```

该语句将存储在`power`变量中的值进行加1操作。

使用`--`运算符可以将变量减1，如下所示：

```
rating--;
```

该语句将变量`rating`中的值减1。

也可以将递增运算符和递减运算符放置在变量名的前面，如下面的语句所示。

```
++power;  
--rating;
```

将运算符放置在变量前面，称之为运算符前置；放到后面，则称之为运算符后置。

By the Way

注意

有点困惑？其实，当你回想一下在小学时学到的前缀的概念，就会发现前置与后置的概念很简单。与`sub-`或`un-`这样放在单词前面的前缀相同，

前置运算符是放置在变量名的前面，而后置运算符则是放置在变量名的后面。

在表达式中使用递增、递减运算符时，一定要注意运算符前置和运算符后置的区别，这很重要。

考虑下面的语句：

```
int x = 3;  
int answer = x++ * 10;
```

在这两条语句处理完毕之后，`answer`变量的值是多少呢？你可能认为应该是40。当然，如果是对3加1，这样等于4，然后再乘以10，这样就是40。

然而，`answer` 变量实际的值却是 30，因为这里使用的后置运算符，而不是前置运算符。

当表达式中的变量使用了后置运算符时，只有当整个表达式执行完毕之后，变量的值才会发生改变。语句`int answer = x++ * 10`的执行顺序与下面的两条语句相同。

```
int answer = x * 10;  
x++;
```

如果使用的是前置运算符，则顺序相反。也就是说，如果表达式的变量中使用的是前置运算符，则该变量的值会在表达式执行完毕之前就发生改变。

来看下面的语句：

```
int x = 3;  
int answer = ++x * 10;
```

此时`answer`变量的值将等于40。前置运算符会在表达式执行之前，先将变量`x`加1。语句`int answer = ++x * 10`的执行顺序与下面的语句相同：

```
x++;  
int answer = x * 10;
```

读者很容易就会对++运算符和--运算符产生恼怒心理，因为它们不像本书中出现的其他概念那样，容易让人理解。

By the Way

注意

第1章介绍C++编程语言的名称时，指出这是一个玩笑的结果，以后读者将明白。知道运算符++，读者就能够明白为什么C++使用两个+而不是

一个。由于C++在C语言的基础上增加了很多新特性和功能，它被看做C的增量，因此将其命名为C++。

在阅读本书章后，读者也能够讲这个笑话，且全世界99%的人都不会理解。

在编写Java程序时，也可以不使用递增和递减运算符，通过如下方式使用+和-运算符，同样也可以实现同样的结果。

```
x = x + 1;  
y = y - 1;
```

递增和递减是很有效的便捷方式，但也会让表达式看起来更长。

5.5.2 运算符优先级

当在一个表达式中使用多个运算符时，需要知道计算机处理这些运算符的先后顺序。来看下面的语句：

```
int y = 10;  
x = y * 3 + 5;
```

在计算机处理这些语句时，除非你事先知道其处理的先后顺序，否则将很难确定x变量的值到底是多少。取决于y*3先执行，还是3+5先执行，x的结果可能是35，也可能是80。

在计算表达式时，将会按照如下顺序来执行。

1. 先执行递增和递减操作。
2. 然后执行乘、除以及取模运算。
3. 然后执行加、减操作。
4. 然后是“比较”操作。
5. 最后使用等号 (=) 来设置变量的值。

由于乘法运算先于加法运算发生，因此可以反过头来看前面的例子，然后得出答案：y先与3相乘，其结果是30，然后再加5，这样x变量的值最终为35。

“比较”操作将在第7章讲解。其他运算顺序已经在本章讲解了，因此现在可以求出下面语句的结果了。

```
int x = 5;  
int number = x++ * 6 + 4 * 10 / 2;
```

上述语句执行完毕之后，number变量的值是50。

计算机是如何得出这个结果来的呢？计算机首先处理递增运算符，因此x++表达式将变量x的值设为6。注意，++运算符位于变量x的

后面，是后置运算符，也就是说，在计算该表达式时，使用的是变量x的初始值。

由于使用的是变量x在递增之前的初始值，因此表达式如下所示：

```
int number = 5 * 6 + 4 * 10 / 2;
```

现在，计算机开始从左向右处理乘除运算。即5先乘以6，然后是4乘以10，这个结果再除以2，这样表达式成为下面的样子：

```
int number = 30 + 20;
```

该表达式将变量number的值设置为50。

如果想以不同的顺序来对表达式求解，可以使用小括号把表达式的一部分括起来，它们就可以先被处理了。例如， $x = 5 * 3 + 2$ ；在执行完毕时，x的值是17，原因就是先执行乘法运算，然后再执行加法运算。然而，如果将该表达式修改为下面的形式：

```
x = 5 * (3 + 2);
```

此时，括号内的表达式将先被执行，这样x的最终结果就是25。读者可以根据需要在语句中使用括号。

5.6 使用表达式

当你小时候在学校遇到令人讨厌的数学题时，是否抱怨过高次乘方，发誓今后再也不使用这样的知识？对不起，这里可能破坏你的誓言，但是你的老师是正确的——这些数学知识将在计算机编程中得到应用。这真是个坏消息。

但好消息是计算机将帮你做这些数学题。计算机程序中经常使用表达式，可以使用它们完成下面这样的任务：

- 修改变量的值；
- 在程序中计数；
- 在程序中使用数学公式。

编写计算机程序并使用表达式时，你又回到了过去的数学课中。表达式可以使用加、减、乘、除和求模。

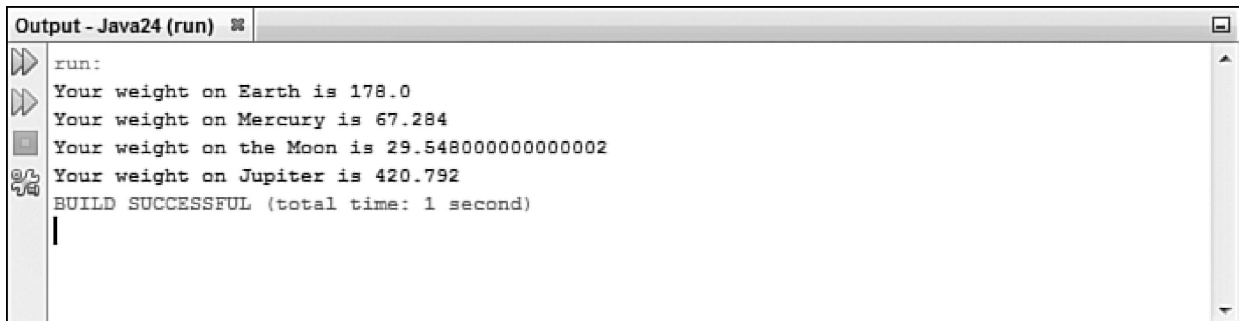
要看一看运行中的表达式，回到 **NetBeans** 并创建一个新的 **Java** 文件，其类名为 **PlaneWeight**。这个程序可以记录某一个人在进入到太阳系时，其体重的增减。在源代码编辑器中输入程序清单5.2中的所有代码。下面依次讲解该程序的每一部分。

程序清单5.2 PlaneWeight程序

```
1: package com.java24hours;
```

```
2:
3: class PlanetWeight {
4:     public static void main(String[] arguments) {
5:         System.out.print("Your weight on Earth is ");
6:         double weight = 178;
7:         System.out.println(weight);
8:
9:         System.out.print("Your weight on Mercury is ");
10:        double mercury = weight * .378;
11:        System.out.println(mercury);
12:
13:        System.out.print("Your weight on the Moon is ");
14:        double moon = weight * .166;
15:        System.out.println(moon);
16:
17:        System.out.print("Your weight on Jupiter is ");
18:        double jupiter = weight * 2.364;
19:        System.out.println(jupiter);
20:    }
21: }
```

输入完毕后，保存文件，它应该会自动编译。然后通过菜单命令 Run->Run File来运行该程序。输出结果如图5.1所示。

The screenshot shows a window titled "Output - Java24 (run)". It contains the following text: "run:", "Your weight on Earth is 178.0", "Your weight on Mercury is 67.284", "Your weight on the Moon is 29.548000000000002", "Your weight on Jupiter is 420.792", and "BUILD SUCCESSFUL (total time: 1 second)". There is a vertical cursor at the end of the last line.

```
run:
Your weight on Earth is 178.0
Your weight on Mercury is 67.284
Your weight on the Moon is 29.548000000000002
Your weight on Jupiter is 420.792
BUILD SUCCESSFUL (total time: 1 second)
```

图5.1 PlaneWeight程序的输出

和你创建的其他程序一样，PlaneWeight程序也使用了 main()块语句来进行相应的处理。该语句可以分成下面4部分。

1. 第5～7行：将人在地球上的初始体重设置为178。
2. 第9～11行：计算人在水星上的体重。
3. 第13～15行：计算人在月球上的体重。
4. 第17～19行：计算人在木星上的体重。

第6行创建了`weight`变量，并使用`double`将其指定为一个较大的浮点型变量。该变量的初始值为178，在整个程序中用于监控人的体重。

下一行类似于程序中的其他语句：

```
System.out.println(weight);
```

命令`System.out.println()`显示其括号中包含的字符串。在第5行，`System.out.print()`命令显示文本“Your weight on Earth is”。程序中还有多个`System.out.print()`和`System.out.println()`语句。

它们之间的区别在于，`print()`在显示完文本后，不会自动换行，而`println()`则会自动换行。

5.7 总结

学习了变量和表达式后，读者将能够在程序中向计算机发出更多的指令。

有了本章学习到的知识，读者可以编写程序，完成众多类似计算器的任务，轻松地处理复杂的数学等式。

数字只是一种可存储到变量中的信息，还可以存储字符、字符串以及称为布尔变量的特殊值true和false。下一章将介绍字符串变量以及他们是如何存储和使用的。

5.8 问与答

问：在Java程序中，一行和一条语句是一回事吗？

答：不是。虽然本书的程序都将每条语句放在一行内，但旨在让程序更容易理解，并非必须如此。

Java编译器在编译程序时不考虑行、空格或其他格式问题，编译器只要求每条语句以分号结束。下面的这一行代码对Java来说也无问题：

```
int x = 12; x = x + 1;
```

可以在一行中放置多条语句，但这样做让阅读源代码的人难以理解。鉴于这种原因，通常不建议这样做。

问：为什么变量名的首字母要小写，如gameOver？

答：这是一种命令约定，而且对你的编程很有帮助。首先，它可以让变量更容易识别，从而与Java程序中的其他元素区分开来。另外，

通过采用统一的变量命名规则，当你在程序的多个地方使用同一个变量时，更容易修复出现的错误。本书使用的大小写规则已被大多数Java程序员采用多年。

问：我可以在Java程序中将二进制值赋值给整型变量么？

答：可以这样做，这是一个在Java的最近版本中引入的特性。在为整型变量赋二进制值时，需要在数值前面添加字符**0b**。由于**1101**是数字**13**的二进制值，因此下面的语句是将**13**赋值给整型变量**z**。

```
int z = 0b0000_1101;
```

下划线的目的只是为了增加数值的可读性，它会被Java编译器忽略。

十六进制值可以使用前导为“**0x**”的数值来表示。比如在第**48**届超级碗NFL总决赛中（**Super Bowl 0x30**），西雅图海鹰队以**43:8**（**0x2B : 0x8**）的比分战胜了丹佛野马队。

5.9 测验

回答下列问题，以测试对变量、表达式以及本章介绍的其他知识的理解程度。

5.9.1 问题

1. 位于左大括号和右大括号之间的一组语句称为什么？

- a. 块语句。
- b. 组件（groupware）。
- c. 用括号括起的语句。

2. 布尔变量用于存储true或false。

- a. 正确。
- b. 错误。
- c. 不，谢谢，我吃过了。

3. 变量名不能以哪些字符打头？

- a. 美元符号（\$）。
- b. 双斜线（//）。
- c. 字母。

5.9.2 答案

1. a. 括起来的语句称为块语句或块。

2. a. 布尔变量只能存储true或false。

3. b. 变量名可以以字母、美元符号 (\$) 或下划线 (_) 打头。如果变量名以两个斜杠打头，该代码行后面的内容将被忽略，因为双斜杠表示注释行。

5.10 练习

为充分复习本章的主题，请完成下面的练习。

- 扩展PlaneWeight程序，记录人在金星上的体重（大约是人在地球上的体重的90.7%），以及人在天王星上的体重（大约是地球上的88.9%）。
- 创建一个简短的Java程序，它使用整型变量x和y，并显示x和y的平方和。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第6章 使用字符串来交流

本章介绍如下内容：

- 使用字符串来存储文本；
- 在程序中显示字符串；
- 在字符串中包含特殊的字符；
- 拼接字符串；
- 在字符串中包含变量；
- 比较字符串；
- 判断字符串的长度。

我们的计算机程序能够安静地工作，从来不停下来聊天。

Java程序将字符串作为与用户交流的主要方式。字符串是一组文本，可以包含字母、数字、标点符号及其他字符。本章将介绍如何在Java程序中使用字符串。

6.1 在字符串中存储文本

字符串用来存储文本并显示给用户。字符串中最基本的元素是字符。字符可以是一个字母、数字、标点符号，或者是其他符号。

在Java程序中，字符是可存储到变量中的信息类型之一，字符型变量是在语句中使用char类型来创建的，如下所示：

```
char keyPressed;
```

这条语句创建了一个名为**keyPressed**的变量，可用于存储字符。当创建字符型变量时，可以设置其初始值，如下所示：

```
char quitKey = '@';
```

字符的值必须用单引号括起来。

字符串是一组字符，可以使用**String**和变量名来创建存储字符串值的变量，如下所示：

```
String fullName = "Fin Shepard"
```

这条语句创建了一个名为**fullName**的字符串变量，并在其中存储了文本“Fin Shepard”，这是2013年的电影《Sharknado》中主人公的名字。在Java语句中，字符串用双引号括起，但双引号不是字符串的一部分。

不同于你前面使用过的其他类型的变量：**int**、**float**、**char**、**boolean**，表示字符串类型的**String**的首字母必须大写。

字符串是一种称为对象的特殊信息，在Java中，所有对象的类型的首字母都必须大写。有关对象的知识将在第10章介绍。在本章需要注意的是，字符串与其他变量类型不同，由于这种差别，在语句中指定字符串类型时，**String**的首字母必须大写。

6.2 在程序中显示字符串

在Java程序中，显示字符串的最基本方法是使用`System.out.println()`语句。该语句可在括号中接收字符串和其他变量，并将它们显示在系统输出设备中，即计算机监视器。下面是一个例子。

```
System.out.println("We can't just wait here for sharks to rain down on us.");
```

上述语句将显示下列文本：

```
We can't just wait here for sharks to rain down on us.
```

在屏幕上显示文本通常称为打印，这就是`println()`代表的意思：打印该行。你可以使用`System.out.println()`语句显示用双引号括起的文本，还可以显示变量（稍后你将会看到）。所有要显示的内容都需要放在括号内。

另一种显示文本的方法是调用`System.out.print()`，该语句显示括号中的字符串和其他变量，但不同于`System.out.println()`，它让接下来的语句在同一行显示文本。

可以连续使用`System.out.print()`多次，将内容显示在同一行，如下例所示：

```
System.out.print("There's ");  
System.out.print("a ");
```

```
System.out.print("shark ");  
System.out.print("in ");  
System.out.print("your ");  
System.out.print("pool. ");
```

这些语句输出的文本如下：

```
There's a shark in your pool.
```

6.3 在字符串中使用特殊字符

创建或显示字符串时，其文本必须用双引号括起。这些双引号不会显示出来，这就提出了一个很好的问题：如果要显示双引号该怎么办呢？

为显示双引号，Java创建了一个特殊编码`\`，可放到字符串中。在字符串中遇到该编码时，将其替换为双引号。例如，请看下面的例子：

```
System.out.println("Anthony Ferrante directed \"Sharknado\" in 2013.");
```

这段代码显示如下内容：

```
Anthony Ferrante directed "Sharknado" in 2013.
```

可以采用这种方式在字符串中插入多个特殊字符，下面列表出了这些特殊字符，注意到每个都是以反斜线（\）打头。

特殊字符	显示
\'	单引号
\"	双引号
\	反斜线
\t	制表符
\b	回退符
\r	回车符
\f	走纸符
\n	换行符

换行符的作用是在下一行行首显示换行符后面的文本，请看下面的例子：

```
System.out.println("Script by\nThunder Levin");
```

这条语句将显示如下内容:

```
Script by  
Thunder Levin
```

6.4 拼接字符串

使用`System.out.println()`语句以及用其他方式处理字符串时, 可以使用加号 (+) 将两个字符串拼接起来。这里用到的加号和用来对数值进行求和的加号相同。

将运算符+用于字符串, 其含义与原来不同: 不是执行数学运算, 而是将两个字符串拼接起来。这导致字符串显示在一起, 或使用两个小字符串组合成一个长字符串。

这种行为用拼接 (concatenation) 来描述, 因为它的意思是将两样东西连接起来。

***By the
Way***

注意

读者在学习编程技巧时, 可能会在其他书中看到concatenation这个术语。但是, 本书在讲解字符串和字符串结合时, 用的是pasting这个术语。

下面的语句使用+运算符来显示一个长字符串:

```
System.out.println("\"'\Sharknado\' is an hour and a half of your ")
```

```
+ "life that you'll never get back.\nAnd you won't want to.\n"\n+ "\t-- David Hinckley, New York Daily News");
```

这里不是将整个字符串放在单独一行（如果这样，以后查看程序时将更难理解），而是使用运算符+将文本文件分成两行。执行这条语句时，输出结果如下：

```
"'Sharknado' is an hour and a half of your life that you'll never get  
back. And you won't want to."  
  -- David Hinckley, New York Daily News
```

在该字符串中使用了几个特殊字符：\、\'、\n和\t。为了更好地熟悉这些字符，请将输出和生成输出的System.out.println()语句进行比较。

6.5 将其他变量用于字符串中

虽然可以使用+运算符将两个字符串拼接起来，但更常见的是使用它将字符串和变量拼接起来。请看下面的例子：

```
int length = 86;  
char rating = 'R';  
System.out.println("Running time: " + length + " minutes");  
System.out.println("Rated " + rating);
```

这段代码的输出如下：


```
Running time: 86 minutes  
Rated R
```

这个例子说明了将+运算符用于字符串的独特之处：导致不是字符串的变量作为字符串显示出来。`length`是一个整型变量，其值为86，它显示在字符串“Running time:”和“minutes”之间。`System.out.println()`语句用于显示一个字符串加上一个整数再加上一个字符串。这条语句之所以能够正常运行，是因为至少开头的部分是字符串。Java语言通过提供这种功能使信息更容易显示。

读者可能想做的一件事情是，将字符串拼接多次，如下例所示：

```
String searchKeywords = "";  
searchKeywords = searchKeywords + "shark ";  
searchKeywords = searchKeywords + "hurricane ";  
searchKeywords = searchKeywords + "danger";
```

这段代码导致变量`searchKeywords`被设置为“shark hurricane danger”。第1行创建变量 `searchKeywords` 并将其设置为空字符串，因为双引号之间为空。第2行将变量 `search Keywords`设置为其当前值加上字符串`shark`；接下来的两行用相同的方式再加上`hurricane`和`danger`。

可以看到，在变量后面拼接文本时，变量名将出现两次。Java提供了一种快捷方式来简化该过程，这就是+=运算符。+=运算符将=和+运算符的功能融为一体。对于字符串，它用于在当前字符串后面加上其他字符串。上述`searchKeywords`示例可以使用+=运算符简化为如下所示：

```
String searchKeywords = "";
searchKeywords += "shark ";
searchKeywords += "hurricane ";
searchKeywords += "danger";
```

这段代码的效果与前面相同：将searchKeywords设置为“shark hurricane danger”。

6.6 字符串的高级处理

还有多种其他方式可用于查看字符串变量和修改其值。之所以有这些高级功能，是因为字符串在Java语言中是对象。通过处理字符串对象获得的知识，也适用于其他对象。

6.6.1 比较两个字符串

在程序中经常要测试两个字符串是否相等，为此可在带有两个字符串的语句之中使用equals()，如下所示：

```
String favorite = "chainsaw";
String guess = "pool cue";
System.out.println("Is Fin's favorite weapon a " + guess + "?");
System.out.println("Answer: " + favorite.equals(guess));
```

这里使用了两个字符串变量，一个是变量favorite，用于存储Fin最喜欢的捕鲨工具——电锯；另一个变量是guess，用于存储对其最喜欢的工具的猜测，该猜测是Fin最喜欢台球杆。

第3行显示文本“Is Fin’s favorite weapon a”，后跟变量`guess`的值和问号。
第4行显示文本“Answer: ”以及下面的新内容：

```
favorite.equals(guess)
```

语句中的这部分称为方法。方法是在Java程序中完成任务的一种方式，这里的方法要完成的任务是，比较字符串`favorite`和字符串`guess`的值是否相等。如果这两个字符串的值相等，就显示`true`，否则显示`false`。下面是该示例的输出结果：

```
Output ▼  
Is Fin's favorite weapon a pool cue?  
Answer: false
```

6.6.2 确定字符串的长度

有时确定字符串的长度很有用，为此可使用方法`length()`。该方法的工作原理与`equals()`相似，但只涉及一个字符串变量。请看下面的例子：

```
String cinematographer = "Ben Demaree";  
int nameLength = cinematographer.length();
```

该示例将整型变量`nameLength`的值设置为11，方法`cinematographer.length()`计算字符串变量`cinematographer`包含的字符数，并将结果赋给整型变量`nameLength`。

6.6.3 改变字符串的大小写

计算机很不灵活，不能识别明显相同的东西。虽然人很容易识别出文本 Ian Ziering 和 IAN ZIERING 是一回事，但大多数计算机不这么认为。例如，在本章前面介绍的 equals() 方法将果断地认为 Ian Ziering 不等于 IAN ZIERING。

为了绕过这些障碍，Java 提供了将字符串变量全部转换为大写的方法（toUpperCase()）和全部转换为小写的方法（toLowerCase()）。下面的例子演示了如何使用方法 toUpperCase()：

```
String fin = "Ian Ziering";  
String change = fin.toUpperCase();
```

这段代码将字符串变量 change 设置为字符串变量 fin 的大写形式，即 IAN ZIERING。toLowerCase() 方法的用法相同，但返回的是字符串的小写。

注意，toUpperCase() 方法不改变它调用的字符串变量的大小写。在上述示例中，变量 fin 的值仍为 Ian Ziering。

By the Way

注意

绕过这一障碍的另一种方法是调用 equalsIgnoreCase() 方法，它在比较两个字符串时不考虑大小写。调用该方法比较字符串 “Tara Reid” 和 “TARA REID” 时，返回的结果为 true，表示它俩匹配。

6.6.4 查找字符串

处理字符串时，另一项常见的任务是，确定在一个字符串中能否找到另一个字符串。要在字符串中查找，可使用方法`indexOf()`，并将要查找的字符串放在括号中。如果没有找到指定的字符串，`indexOf()`返回-1；如果找到，`indexOf()`返回一个整数，指出该字符串的起始位置。字符串中字符位置从0开始编号，即第一个字符的位置为0。在字符串“Sharknado”中，文本“nado”的起始位置为5。

`indexOf()`方法的一种用法是搜索电影《Sharknado》的整个剧本，找到主人公将直升飞机开入到龙卷风中投弹，并且Nova说“We’re gonna need a bigger chopper”的地方。

如果《Sharknado》的整个剧本存储在变量`script`中，可以使用下面的语句从中搜索上面这句话：

```
int position = script.indexOf("We're gonna need a bigger chopper");
```

如果在`script`字符串中找到该文本，变量`position`将等于该文本在`script`中的起始位置，否则将等于-1。

如果你打算在一个字符串中寻找另外一个字符串，但是不关心其位置，可以使用字符串的`contains()`方法，它将返回一个布尔值。如果找到了字符串，返回`true`；否则返回`false`。下面是一个示例：

```
if (script.contains("There's a shark in your pool")) {  
    int stars = 4;  
}
```

Watch Out!

警告:

`indexOf()`和`contains()`方法是区分大小写的，这也就意味着只有当目标字符串和搜索字符串的大小写完全相同时，才算查找成功。否则，`indexOf()`返回-1，`contains()`返回false。

6.7 导演及演员名单

接下来，为了加深读者对前面介绍的字符串处理功能的理解，将编写一个Java程序，显示一部电影的导演和演员名单。你应该能够猜到该电影的名字。

返回NetBeans中的Java24项目，然后在`com.java24hours`包中创建一个名为Credits的Java空文件，在源代码编辑器中输入程序清单6.1中的所有文本，输入完毕之后存盘。

程序清单6.1 Credits程序

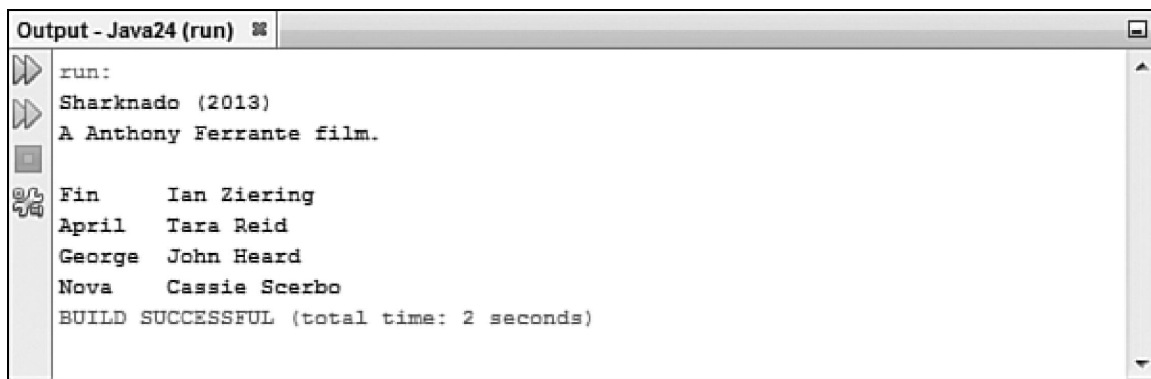
```
1: package com.java24hours;
2:
3: class Credits {
4:     public static void main(String[] arguments) {
5:         // set up film information
6:         String title = "Sharknado";
7:         int year = 2013;
8:         String director = "Anthony Ferrante";
9:         String role1 = "Fin";
10:        String actor1 = "Ian Ziering";
11:        String role2 = "April";
12:        String actor2 = "Tara Reid";
13:        String role3 = "George";
14:        String actor3 = "John Heard";
15:        String role4 = "Nova";
16:        String actor4 = "Cassie Scerbo";
```

```
17:         // display information
18:         System.out.println(title + " (" + year + ")\n" +
19:             "A " + director + " film.\n\n" +
20:             role1 + "\t" + actor1 + "\n" +
21:             role2 + "\t" + actor2 + "\n" +
22:             role3 + "\t" + actor3 + "\n" +
23:             role4 + "\t" + actor4);
24:     }
25: }
```

编译程序前先浏览一遍程序，看是否能够明白各条语句的功能。对该程序的详细分析如下。

- 第3行将该Java程序命名为Credits。
- 第4行是main() 块语句的开头，程序的所有功能都是在该块语句中完成的。
- 第6～16行创建用于存储导演和演员以及影片信息的变量。其中一个变量year，它是一个整型变量，其他变量都是字符串变量。
- 第18～23行是长语句System.out.println()。在第18行和第23行的括号之间的信息都将显示到屏幕上。换行符\n的作用是将其后面的文本在下一行的行首显示。制表符\t的作用是在输出信息中插入制表符。其他要显示的内容要么是文本，要么是字符串变量。
- 第24行结束main()块语句。
- 第25行结束整个程序。

如果提示有错误，可以修改Credits程序中的任何输入错误，然后重新保存。NetBeans将自动编译程序。当运行程序时，将会看到如图6.1所示的输出窗口。



```
Output - Java24 (run)
run:
Sharknado (2013)
A Anthony Ferrante film.
Fin    Ian Ziering
April  Tara Reid
George John Heard
Nova   Cassie Scerbo
BUILD SUCCESSFUL (total time: 2 seconds)
```

图6.1 Credits程序的输出

6.8 总结

如果你的Credits程序的输出结果与图6.1相同，你的信心也应该增加。通过本书前6章的学习，你现在可以编写一个比较长的Java程序，而且也可以处理一些比较复杂的问题了。字符串是你在每次编程时都会用到的东西。你可以通过多种方式使用字符串来与用户交流。

6.9 问与答

问：如何将字符串变量的值设置为空？

答：一对双引号之间没有任何文本就表示空字符串。下面的代码创建一个名为georgeSays的字符串变量，并将其值设置为空：

```
String georgeSays = "";
```

问：使用 toUpperCase() 方法好像不能将字符串中的字母全部转换为大写，我哪里操作不正确？

答： 调用字符串对象的`toUpperCase()`方法时，实际上它并未修改该字符串对象，而是创建一个字母全部大写的新字符串，请看下面的语句：

```
String firstName = "Baz";  
String changeName = firstName.toUpperCase();  
System.out.println("First Name: " + firstName);
```

这些语句的输出结果为“First Name: Baz”，因为变量`firstName`包含的是原来的字符串。如果将最后一条语句改为显示变量`changeName`，输出结果将为“First Name: BAZ”。

当字符串在Java中创建之后，它们的值不会发生改变。

问： 在 Java 中，就字符串而言，所有方法都像 `equals()` 那样返回 `true` 或 `false` 吗？

答： 方法被调用后，可以有不同的方式来进行响应。如果方法像 `equals()` 那样发回一个值，则被称为返回一个值。方法 `equals()` 返回一个布尔值，其他方法可能返回字符串、整数、其他类型的值，也可能什么都不返回（使用 `void` 来表示）。

6.10 测验

回答下列问题，以测试对字符串理解和掌握的程度。

6.10.1 问题

1. 我的朋友要执行拼接操作，需要向权威部门报告吗？

- a. 不，仅在冬季这才是非法的。
 - b. 是的，但要等到我将故事卖给TMZ.com再说。
 - c. 不，他所做的只是在程序中将两个字符串连接起来。
2. 为什么String的首字母要大写，而int等类型名的首字母不需要大写？
- a. String是一个完整的单词，而int不是。
 - b. 和Java中所有的对象一样，String的首字母必须大写。
 - c. Oracle的质量控制做得很糟糕。
3. 下列那项在字符串中添加一个单引号？
- a. <quote>。
 - b. \'。
 - c. ‘。

6.10.2 答案

1. c. 拼接（Concatenation）指的是将两个字符串连接起来，它使用运算符+或+=。
2. b. 在Java中，所有的对象类型名的首字母都要大写，因此变量名的首字母都是小写，这样就不容易将变量和对象搞混。
3. b. 在字符串中插入特殊字符时，总是以单个反斜杠开头。

6.11 练习

通过下列练习来复习本章介绍的主题。

- 编写一个名为Favorite的小型Java程序，将本章中“比较两个字符串”一节的代码放在main()块语句中。测试该程序，确保其输出就像正文中描述的那样，Fin最喜欢的捕鲨工具不是台球杆。测试完毕后，将变量guess的初始值从pool cue改为chainsaw，再看看结果如何。
- 修改程序Credits，将导演和全部演员的名字都用大写字母显示。

有关完成这些练习需要编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第7章

使用条件测试进行判断

本章介绍如下内容：

- 使用if语句进行基本的条件测试；
- 测试一个值是大于还是小于另一个值；
- 测试两个值是否相等；
- 使用与if语句对应的else语句；
- 组合多个条件测试；
- 使用switch语句进行复杂的条件测试；
- 使用三元运算符创建测试。

编写计算机程序时，你提供给计算机的是一系列称为语句的指令，这些指令被严格地执行。你可以让计算机计算令人讨厌的数学公式，它将为你得出结果；也可以让计算机显示一些信息，它将忠实地完成。

然而，有时需要让计算机做出选择。例如，编写计算账目收支平衡的程序时，可能想在账户透支时让计算机显示一条警告消息，而且只有当账户透支时，计算机才显示该消息；如果没透支也显示这样的消息，计算机程序就不准确，也让人感到不安。

在Java程序中，完成这项工作的方法是使用条件语句。而且仅当满足特定的条件时，条件语句才导致特定的操作。本章将介绍如何使用各种条件语句：if、else和switch。

在Java程序中，决策都是使用条件语句来完成的。在本章，读者将在Java程序中使用条件关键字if、else、switch、case、break检查各种条件，还将使用多种条件运算符，如==、!=、<、>和?，以及布尔变量。

7.1 if语句

在Java程序中，对条件进行测试所使用的最基本的方法是if语句。if语句测试某个条件为true还是false，并在条件为true时执行特定的操作。

使用if和条件来进行测试，如下面的语句所示：

```
long account = -17_000_000_000_000L;
if (account < 0) {
    System.out.println("Account overdrawn; you need a bailout");
}
```

if语句使用小于运算符<来判断account变量是否小于0。如果是，则运行if语句中的代码块，然后显示文本。

只有条件为真时，if语句中的代码块才运行。在上面的例子中，如果变量account的值大于或等于0，println语句将不执行。注意，在if语句中测试的条件要用括号括起，如(account < 0)。

小于运算符 (<) 是可以在条件语句中使用的多种运算符之一。

7.1.1 小于和大于的比较

在前一节，像在数学课中那样使用了运算符 (<)：表示小于。还有大于运算符 (>)，下面的语句演示了如何使用该运算符：

```
int elephantWeight = 900;
int elephantTotal = 13;
int cleaningExpense = 200;
if (elephantWeight > 780) {
    System.out.println("Elephant too big for tightrope act");
}
if (elephantTotal > 12) {
    cleaningExpense = cleaningExpense + 150;
}
```

第一条if语句测试变量elephantWeight的值是否大于780，第二条if语句测试变量elephantTotal的值是否大于12。

如果上述语句位于这样的程序中，即变量elephantWeight等于600，而变量elephantTotal等于10，则if语句块中的语句将不会执行。

有时可能需要判断一个变量是否小于或等于某个值，为此需要使用运算符<=。下面是一个例子：

```
if (account <= 0) {
    System.out.println("You are flat broke");
}
```

还有一个>=运算符可以用于“大于或等于”测试。

7.1.2 相等和不等

在程序中要检测的另一种条件是否相等。变量是否等于特定的值？一个变量的值是否与另一个变量的值相等？这些问题可以使用运算符==来回答，如下面的语句所示：

```
if (answer == rightAnswer) {  
    studentGrade = studentGrade + 10;  
}  
if (studentGrade == 100) {  
    System.out.println("Show off!");  
}
```

Watch Out!

警告：

用于检测是否相等的运算符由两个等号组成，即==。很容易将其同运算符=混淆，后者用于给变量赋值。在条件语句中，总是使用两个等号。

读者也可以测试不相等，即一个值是否不等于另一个值，为此可使用运算符!=，如下例所示：

```
if (answer != rightAnswer) {  
    score = score - 5;  
}
```

除字符串变量之外（因为字符串是对象），可以将运算符==和!=用于任何类型的变量。

7.1.3 使用块语句组织程序

到目前为止，所有的if语句后面都有一个代码块，该代码块位于{ }之间（这两个字符的技术术语应该是“大括号”）。

在本书前面，读者看到过如何使用块语句来标识Java程序中main()语句块的起始和结束。程序运行时，main()语句块中的每条语句都被执行。

if语句不需要块语句，它只占用单独的一行，如下面的例子所示：

```
if (account <= 0) System.out.println("No more money");
```

只有当if条件中的条件为true时，if条件后的语句才能执行。

程序清单7.1的Java程序使用块语句来表示main()块。该块语句从第3行的左大括号“{”开始，到第15行的右大括号“}”结束。打开NetBeans，在com.java24hours包中创建一个空的Java文件，命名为Game，然后输入程序清单7.1中的所有文本。

程序清单7.1 Game程序


```
1: package com.java24hours;
2:
3: class Game {
4:     public static void main(String[] arguments) {
5:         int total = 0;
6:         int score = 7;
7:         if (score == 7) {
8:             System.out.println("You score a touchdown!");
9:         }
10:        if (score == 3) {
11:            System.out.println("You kick a field goal!");
12:        }
13:        total = total + score;
14:        System.out.println("Total score: " + total);
15:    }
16: }
```

运行该程序时，得到的输出如图7.1所示。

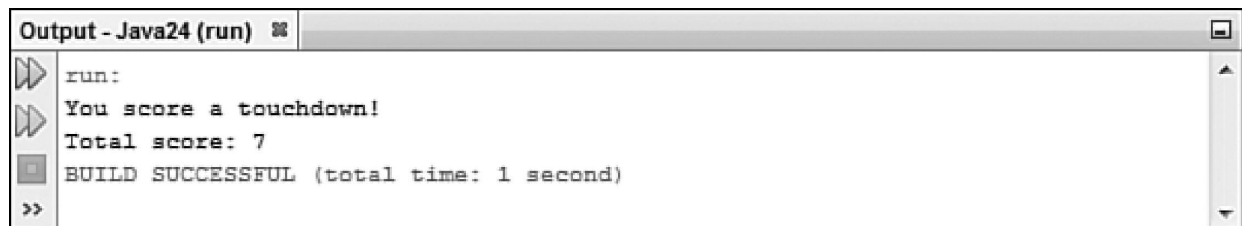


图7.1 Game程序的输出结果

也可以在if语句中使用块语句来让计算机在条件为true时执行多个操作。下面是一条包含块语句的if语句：

```
int playerScore = 12000;
int playerLives = 3;
int difficultyLevel = 10;
if (playerScore > 9999) {
    playerLives++;
    System.out.println("Extra life!");
}
```

```
}    difficultyLevel = difficultyLevel + 5;  
}
```

大括号用于将所有属于if语句的语句编组，如果变量playerScore大于9999，将做3件事情：

- 将变量playerLives的值加1（因为使用了递增运算符++）；
- 显示文本“Extra life!”；
- 将变量difficultyLevel的值加5。

如果变量playerScore不大于9999，将什么都不做，if语句块中的3条语句都将被忽略。

7.2 if-else语句

有时可能要在条件为true时做某些事情，而在条件为false时做另一些事情，为此可以结合使用if语句和else语句，如下例所示：

```
int answer = 17;  
int correctAnswer = 13;  
if (answer == correctAnswer) {  
    score += 10;  
    System.out.println("That's right. You get 10 points");  
} else {  
    score -= 5;  
    System.out.println("Sorry, that's wrong. You lose 5 points");  
}
```

不同于if语句，else语句不包含条件。else语句与前面离它最近的if语句相匹配。也可以使用else语句将多条if语句连接起来，如下所示：

```
if (grade == 'A') {
    System.out.println("You got an A. Awesome!");
} else if (grade == 'B') {
    System.out.println("You got a B. Beautiful!");
} else if (grade == 'C') {
    System.out.println("You got a C. Concerning!");
} else {
    System.out.println("You got an F. You'll do well in Congress!");
}
```

通过这种方式将几个不同的if和else语句放在一起，可以处理很多条件。在上面的示例中，对A类学生、B类学生、C类学生和未来的议员分别发送不同的消息。

7.3 switch语句

if和else语句非常适合于只有两种情况的情形，但有时候需要考虑两种以上的情况。

从前面有关成绩的示例中可知，可以结合使用if和else语句处理多种不同的情况。

另一种方法是使用switch语句。使用switch可以测试多个不同的条件并做出相应的响应。在下面的例子中，使用switch语句对前面有关成绩的示例进行了重写：

```
switch (grade) {  
    case 'A':  
        System.out.println("You got an A. Awesome!");  
        break;  
    case 'B':  
        System.out.println("You got a B. Beautiful!");  
        break;  
    case 'C':  
        System.out.println("You got a C. Concerning!");  
        break;  
    default:  
        System.out.println("You got an F. You'll do well in  
Congress!");  
}
```

第1行的switch语句指定了要检测的变量，在本例中是`grade`，然后switch语句使用“{”和“}”形成了一个块语句。

switch语句中的每条case语句检查变量是否等于某个值，在case语句中使用的值可以是字符、整数或字符串。在这里，这些case语句中使用的值分别是字符‘A’、‘B’和‘C’。每条case语句后跟一条或两条语句。当某条case语句与switch语句中变量的值匹配时，计算机将处理其后面的语句，直到遇到break语句。

例如，如果变量`grade`的值为B，将显示文本“You got a B. Good work!”。接下来是break语句，因此不会执行switch语句的其他部分。break语句告诉计算机退出switch语句。

在switch语句中如果忘记了break语句，将导致不想要的结果。在上面这个例子中，如果没有break语句，则无论`grade`变量是否等于‘A’、‘B’或‘C’，都将显示前三个“You got a”消息。

`default`语句用于处理所有`case`语句都不满足的情况。在这个例子中，如果变量`grade`不等于‘A’、‘B’或‘C’，将进入`default`语句。在程序中，并非在每个`switch`语句块中都需要使用`default`语句。如果没有`default`语句，且所有`case`语句条件都不满足，将什么也不做。

程序清单7.2中的`Commodity`类使用`switch`语句来购买或销售商品（这里没有指明商品到底是什么）。该商品在购买时的价格为`balance-20`美元，卖出时的价格为`balance+15`美元。

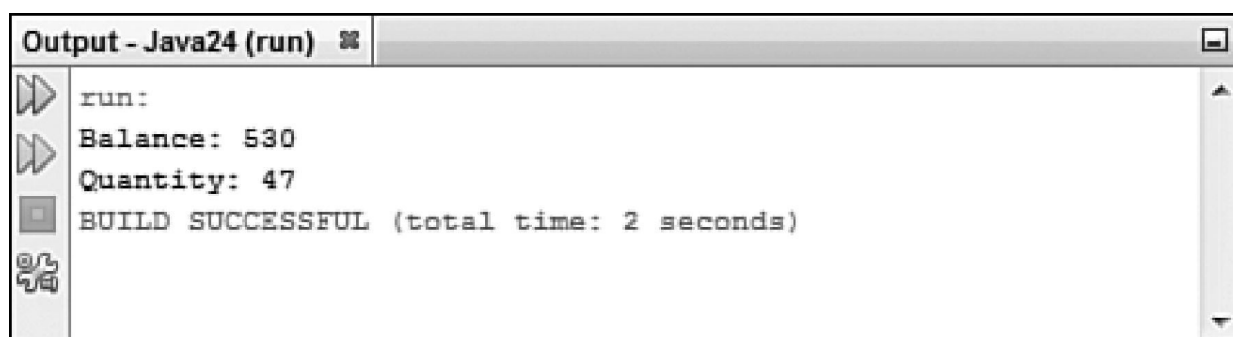
`switch-case`语句对名为`command`的字符串值进行测试，如果等于“BUY”，则运行一个语句块；如果等于“SELL”，则运行另一个语句块。

程序清单7.2 `Commodity`程序

```
1: package com.java24hours;
2:
3: class Commodity {
4:     public static void main(String[] arguments) {
5:         String command = "BUY";
6:         int balance = 550;
7:         int quantity = 42;
8:
9:         switch (command) {
10:             case "BUY":
11:                 quantity += 5;
12:                 balance -= 20;
13:                 break;
14:             case "SELL":
15:                 quantity -= 5;
16:                 balance += 15;
17:         }
18:         System.out.println("Balance: " + balance + "\n"
19:             + "Quantity: " + quantity);
20:     }
21: }
```

在该程序的第5行，`command`字符串被设置为“BUY”。当测试`switch`语句时，将运行第11～第13行的`case`语句块。商品的`quantity`变量增加5，`balance`变量降低20。

当运行`Commodity`程序时，将产生如图7.2所示的输出。



The screenshot shows a window titled "Output - Java24 (run)". The output text is as follows:

```
run:
Balance: 530
Quantity: 47
BUILD SUCCESSFUL (total time: 2 seconds)
```

图7.2 `Commodity`程序的输出

7.4 三元运算符

在Java中，最复杂的条件语句是三元运算符（`?`）。

当你想根据条件测试的结果进行赋值或显示时，可以使用三元运算符。例如，在视频游戏中，可能需要根据变量`skillLevel`是否大于5来设置变量`numberOfEnemies`的值。为此，一种方法是使用`if-else`语句：

```
if (skillLevel > 5) {
    numberOfEnemies = 20;
} else {
    numberOfEnemies = 10;
}
```

另一种快捷方法是使用三元运算符（?）。三元运算符由5部分组成：

- 要测试的条件，用括号括起来，如(skillLevel > 5)；
- 问号（?）；
- 用于判断条件是否为true的值；
- 冒号（:）；
- 用于判断条件是否为false的值。

要使用三元运算符基于skillLevel来设置numberOfEnemies的值，可以使用如下语句：

```
int numberOfEnemies = (skillLevel > 5) ? 20 : 10;
```

也可以使用三元运算符来确定要显示的信息。假如要编写这样一个程序：根据变量 gender的值显示文本“Mr.”或“Ms.”。为此，可以使用三元运算符，如下所示：

```
System.out.print( (gender.equals("male")) ? "Mr." : "Ms." );
```

三元运算符很有用，但也是Java中最难以理解的条件运算符。学习Java时，你不会遇到只能使用三元运算符，而不能使用if-else语句的情况。

7.5 观察时钟

本章的最后一个项目将会为读者展示编程中会用到的每一个条件测试。在这个项目中，将使用Java内置的计时功能，它跟踪当前的日期和时间，并将该信息用一句话显示出来。

运行NetBeans（或其他可以创建Java程序的软件），在com.java24hours中新建一个文件，将其命名为Clock.java。这个程序很长，但大部分都是很长的条件语句。在源代码编辑器中输入程序清单7.3中的所有文本，完成后将文件保存。

程序清单7.3 Clock程序

```
1: package com.java24hours;
2:
3: import java.time.*;
4: import java.time.temporal.*;
5:
6: class Clock {
7:     public static void main(String[] arguments) {
8:         // get current time and date
9:         LocalDateTime now = LocalDateTime.now();
10:        int hour = now.get(ChronoField.HOUR_OF_DAY);
11:        int minute = now.get(ChronoField.MINUTE_OF_HOUR);
12:        int month = now.get(ChronoField.MONTH_OF_YEAR);
13:        int day = now.get(ChronoField.DAY_OF_MONTH);
14:        int year = now.get(ChronoField.YEAR);
15:
16:        // display greeting
17:        if (hour < 12) {
18:            System.out.println("Good morning.\n");
```



```
19:     } else if (hour < 17) {
20:         System.out.println("Good afternoon.\n");
21:     } else {
22:         System.out.println("Good evening.\n");
23:     }
24:
25:     // begin time message by showing the minutes
26:     System.out.print("It's");
27:     if (minute != 0) {
28:         System.out.print(" " + minute + " ");
29:         System.out.print( (minute != 1) ? "minutes" :
30:             "minute");
31:         System.out.print(" past");
32:     }
33:
34:     // display the hour
35:     System.out.print(" ");
36:     System.out.print( (hour > 12) ? (hour - 12) : hour );
37:     System.out.print(" o'clock on ");
38:
39:     // display the name of the month
40:     switch (month) {
41:         case 1:
42:             System.out.print("January");
43:             break;
44:         case 2:
45:             System.out.print("February");
46:             break;
47:         case 3:
48:             System.out.print("March");
49:             break;
50:         case 4:
51:             System.out.print("April");
52:             break;
53:         case 5:
54:             System.out.print("May");
55:             break;
56:         case 6:
57:             System.out.print("June");
58:             break;
59:         case 7:
60:             System.out.print("July");
61:             break;
62:         case 8:
63:             System.out.print("August");
64:             break;
65:         case 9:
66:             System.out.print("September");
```

```
67:         break;
68:     case 10:
69:         System.out.print("October");
70:         break;
71:     case 11:
72:         System.out.print("November");
73:         break;
74:     case 12:
75:         System.out.print("December");
76:     }
77:
78:     // display the date and year
79:     System.out.println(" " + day + ", " + year + ".");
80: }
81: }
```

Watch **Out!**

警告:

由于该程序使用了Java 8的新特性，因此如果当前的NetBeans项目被设置为使用较早的Java版本，则不会进行编译。为了确保选择了正确的设置，可以选择File->Project Properties。在Project Properties对话框中，查找Source/Binary Format的值。它应该是JDK 8。

在程序保存之后，仔细阅读程序，来理解程序中条件测试是如何使用的。

除了第3~第4行和第8~第14行外，Clock程序涉及的知识在本章都介绍过。在创建一系列变量用于存储当前日期和时间后，使用一系列if和switch语句来决定显示什么信息。

该程序包含几条`System.out.println()`语句和`System.out.print()`语句，用于显示字符串。

第8~14行使用了名为`now`的`LocalDateTime`变量，变量类型名`LocalDateTime`的首字母必须大写，这是因为`LocalDateTime`是一种对象。

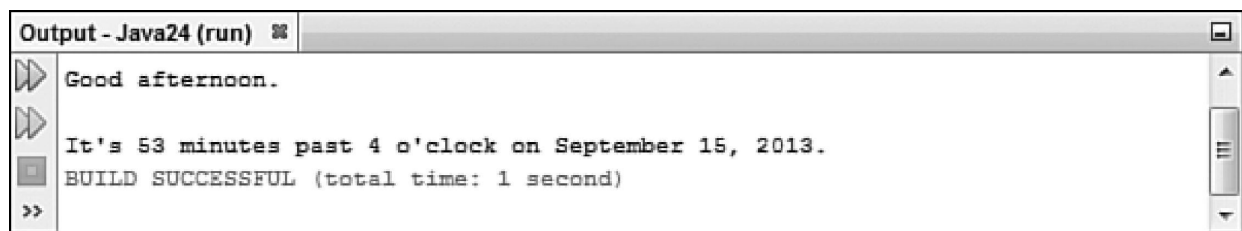
第10章将介绍如何创建和使用对象。这里的重点是第8~14行发生了什么，而不是这是如何发生的。

`Clock`程序由下面几部分组成。

- 第3行让程序能够使用类`java.time.LocalDateTime`，它用于跟踪当前的日期和时间。
- 第4行让程序使用类`java.time.temporalfield.ChronoField`。
- 第6~7行开始`Clock`程序及其`main()`语句块。
- 第9行创建一个名为`now`的`LocalDateTime`对象，该对象包含系统的当前日期和时间。每次运行该程序时，对象`now`的值都不同（当然，除非宇宙的物理定律发生变化或时间停止不前）。
- 第10~14行创建变量`hour`、`month`、`day`和`year`，这些变量的值来自`LocalDateTime`对象，后者是存储所有这些信息的仓库。括号内的信息（比如`ChronoField.DAY_OF_MONTH`）表示要使用日期和时间的哪一部分。
- 第17~23行显示三个问候语之一：“Good morning.”、“Good afternoon.”和“Good evening.”。显示的问候语取决于变量`hour`的值。

- 第26～32行显示当前的分钟值及其他一些文本。首先，第26行显示文本“It’s”，如果minute的值为0，则由于第27行的if语句，第28～31行将不执行。第27行的语句是必需的，因为在程序中告诉人们现在是几小时零分没有意义。第28行显示变量minute的当前值，在第29～30行使用三元运算符显示文本“minutes”或“minute”，这取决于变量minute是否等于1。最后，在第31行显示文本“past”。
- 第35～37行使用另外一个三元运算符显示当前的hour值。第36行的三元运算符条件语句用于在hour的值大于12时以不同的方式显示hour的值，从而避免计算机显示现在是15点这样的情况发生。
- 第40～76行几乎占据了程序的一半，这是一个非常长的switch语句，它根据变量month的值来显示不同的月份名称。
- 第79行显示当前的日期和年份。
- 第80～81行结束main()语句块，然后结束整个Clock程序。

运行该程序时，输出的具体内容随当前日期和时间而异。该程序的输出示例如图7.3所示。



```
Output - Java24 (run)
>> Good afternoon.
>> It's 53 minutes past 4 o'clock on September 15, 2013.
BUILD SUCCESSFUL (total time: 1 second)
```

图7.3 Clock程序的输出

运行该程序多次，看看它如何跟踪时钟变化。

By the
Way

注意

Clock程序使用了Java 8中新引入的Date/Time API。早期的Java版本中使用了不同的类来处理日期和时间。我们曾经在第4章中讲到，Java类库包含数千个执行有用任务的类。本程序中使用的java.time和java.time.temporal包是Date/Time API的一部分。

7.6 总结

能够使用条件语句后，你的Java编程水平将得到极大的提高。即使程序运行时外界信息发生变化，程序也能够对信息进行评测，并据此做出不同的反应。它们可以根据特定的条件，在两个或多个选项之间做出选择。

编写计算机程序时，必须将任务分成一组需要执行的步骤和需要做出的决策。通过在程序中使用if语句和其他条件语句，还可能提高逻辑思维能力，这对生活中的其他方面也有帮助：

- 如果那个候选人在11月份赢得总统选举，我将谋求参与内阁，否则我将移民加拿大；
- 如果我对相亲感到很满意，就在一家豪华餐厅用晚餐，否则将去吃披萨；
- 如果我在试用期犯了错误，则只有奥克兰突袭者队会挑选我。

7.7 问与答

问：if语句好像是最有用的语句之一，有没有这种可能：在程序中只使用if语句，而不用使用其他语句？

答：这是可能的，不使用else和switch，很多程序员也从不使用三元运算符（?）。然而，在程序中使用else和switch是有益的，这样程序更易理解。将一系列if语句放在一起将使程序难以理解。

问：在本章中，将if语句同一条语句结合使用时，没有使用左大括号和右大括号。不是必须使用这对大括号吗？

答：不是的。大括号用于if语句旨在标识根据条件测试结果执行的程序部分。使用大括号是个好习惯，这样可以避免修改程序时犯常见的错误。如果在if条件后面添加第二条语句但没有添加大括号，程序运行时将出现非预期的错误。

问：是不是在每条case语句后面的语句部分都必须使用break语句？

答：不是必须这样做。但如果没有在一组语句末尾使用break，则不管测试出的case值是多少，switch语句块中所有后续语句都将被执行。

然而，大多数情况下，你会希望在每一组语句的后面使用break语句。

7.8 测验

回答下列问题，以检测对Java中条件语句的掌握程度。

7.8.1 问题

1. 条件测试的结果为true或false，这让你想起了哪种变量？
 - a. 没有。不要没完没了地提问。
 - b. long变量。
 - c. 布尔变量。
2. 在switch语句块中，哪条语句用于处理其他所有情况？
 - a. default。
 - b. otherwise。
 - c. onTheOtherHand。
3. 什么是条件？
 - a. 洗发后修理零乱并纠缠在一起的头发。
 - b. 在程序中测试条件是true还是false。
 - c. 向神父坦白罪过的地方。

7.8.2 答案

1. c. 布尔变量只能为true或false，它与条件测试类似。
2. a. 如果没有其他case语句与switch变量匹配，default语句将被执行。

3. b. 其他两项说的是护发素和忏悔。

7.9 练习

为提高对Java条件的认识，通过下面的练习复习本章的主题。

- 在Clock程序某行的break语句前面添加“//”，使其成为注释，然后编译并运行它，看看将发生什么情况。对更多的break语句进行相应处理，再看看发生的情况。
- 创建一个小程序，它在一个名为grade的变量中存储用户从1~100之间选择的值。在条件语句中使用grade变量向A、B、C、D和F类学生显示不同的消息。先用if语句完成这项工作，然后再用switch语句完成。

要查看完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第8章 使用循环重复执行操作

本章介绍如下内容：

- 使用for循环；
- 使用while循环；
- 使用do-while循环；
- 提早退出循环；
- 为循环命名。

对学生来说，最讨厌的惩罚是在纸上或黑板上一遍又一遍地写东西。在电影The Simpsons（辛普森一家）中，Bart Simpson得经常到黑板上写“Organ transplants are best left to professionals”。这种惩罚对孩子也许有效，但对计算机完全无用，因为计算机可以轻松重复一项任务。

计算机程序最适合重复地做相同的事情，因为有循环。循环是在程序中重复执行的一条语句或一组语句。有些循环执行固定的次数，有些循环则可以无限期地执行。

Java中有3种循环语句：for、do和while。这3个循环语句的工作方式相似，但是了解其各自的工作机制仍然大有裨益。通过选择合适的循环语句，可以简化程序的循环部分。

8.1 for循环

在编程时，你会发现循环可以在多种情况下使用。你可以使用循环重复做某些事情很多次，例如防病毒程序打开每封邮件时检查是否有病毒。也可以使用循环让计算机在某一个简短的周期内什么都不做，比如每隔一分钟显示一次当前时间的动态时钟。

循环语句让计算机程序多次返回到同一个地方，就像飞机特技在空中表演转圈时那样。

Java中最复杂的循环语句是for。for循环经常用于重复执行程序某部分特定次数。下面是一个例子。

```
for (int dex = 0; dex < 1000; dex++) {  
    if (dex % 12 == 0) {  
        System.out.println("#: " + dex);  
    }  
}
```

这个循环显示0~999之间可被12整除的数字。

每个for循环都使用一个变量来确定循环何时开始、何时结束。这个变量通常称为计数器（或索引），在上述循环中计数器为变量dex。

这个示例演示了for语句的3部分。

- 初始化部分：在第一部分，变量dex被赋予初始值0。
- 条件部分：在第二部分，有类似于if语句中的条件测试，这里的测试为dex<1000。

- 修改部分：第三部分是一条语句，它使用递增运算符修改变量dex的值。

在初始化部分，可以设置计数器变量，也可以在for语句中创建该变量，就像上例中的整型变量dex那样。当然也可以在程序的其他地方创建该变量。不管是哪种情况，在for语句的这部分都必须给该变量赋初值。循环开始时，变量应该就具备了初始值。

条件部分包含一个测试，要继续循环，该测试的结果必须为true。一旦测试结果为false，循环将结束。在这个例子中，当变量dex的值大于或等于1000时，循环将结束。

for语句的最后一部分包含一条修改计数器变量值的语句。每次循环时，这条语句都将执行。计数器变量必须以某种方式改变其值，否则循环永远都不会结束。在这个例子中，在修改部分，使用运算符++将变量dex的值加1。如果变量dex的值不变，它将始终为初始值0，这样条件dex<1000将永远为true。

for语句中的语句块在每次循环时也都被执行。

在上述例子中，for语句中的语句块如下：

```
if (dex % 12 == 0) {  
    System.out.println("#: " + dex);  
}
```

这条语句将执行1000次。该循环首先将变量dex设置为0，然后每次循环时将变量dex的值加1，当变量dex的值大于或等于1000时循环结束。

By the Way

注意

一个与循环相关的不常见的术语是迭代，它指的是执行单次循环，用于控制循环的计数器变量就是迭代。

在if语句中已经看到，如果for循环只包含一条语句，可以不使用大括号，如下例所示：

```
for (int p = 0; p < 500; p++)  
    System.out.println("I will not sell miracle cures");
```

这个循环显示文本“I will not sell miracle cures”500次。虽然循环只有一条语句时可以不使用大括号，但为使程序易于理解，也可以使用括号，如下所示：

```
for (int p = 0; p < 500; p++) {  
    System.out.println("I will not sell miracle cures");  
}
```

本章创建的第一个程序将显示1~200的整数与9的乘积：即9×1、9×2、9×3等，直到9×200。在NetBeans中，创建一个新的Java空文件，并命名为Nines，然后输入程序清单8.1中的文本。当对文件进行保存时，将存储为Nines.java。

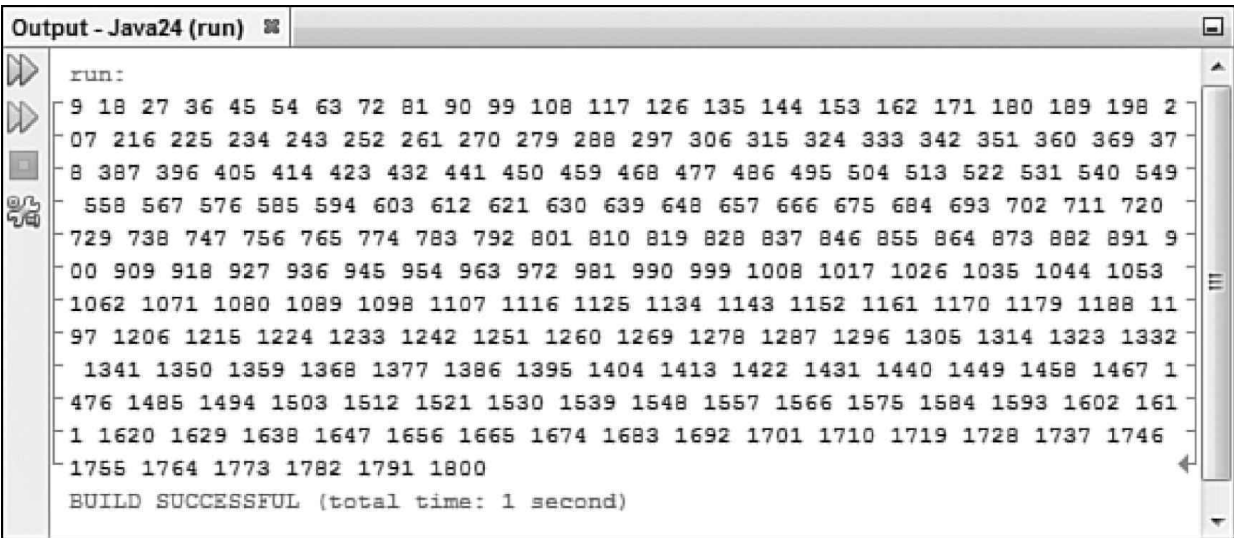
程序清单8.1 Nines.java的完整版本

```
1: package com.java24hours;
2:
3: class Nines {
4:     public static void main(String[] arguments) {
5:         for (int dex = 1; dex <= 200; dex++) {
6:             int multiple = 9 * dex;
7:             System.out.print(multiple + " ");
8:         }
9:         System.out.println();
10:    }
11: }
```

程序Nines的第5行包含一条for语句，该语句有3部分。

- 初始化部分int dex=1：它创建一个整型变量dex，并将其初始值设置为1。
- 条件部分dex <= 200：它必须为真才执行循环，否则将结束循环。
- 修改部分dex ++：在每次循环中将变量dex的值加1。

在NetBeans中选择命令菜单Run->Run File，运行该程序，将产生如图8.1所示的输出。



```
Output - Java24 (run)
run:
9 18 27 36 45 54 63 72 81 90 99 108 117 126 135 144 153 162 171 180 189 198 2
07 216 225 234 243 252 261 270 279 288 297 306 315 324 333 342 351 360 369 37
8 387 396 405 414 423 432 441 450 459 468 477 486 495 504 513 522 531 540 549
558 567 576 585 594 603 612 621 630 639 648 657 666 675 684 693 702 711 720
729 738 747 756 765 774 783 792 801 810 819 828 837 846 855 864 873 882 891 9
00 909 918 927 936 945 954 963 972 981 990 999 1008 1017 1026 1035 1044 1053
1062 1071 1080 1089 1098 1107 1116 1125 1134 1143 1152 1161 1170 1179 1188 11
97 1206 1215 1224 1233 1242 1251 1260 1269 1278 1287 1296 1305 1314 1323 1332
1341 1350 1359 1368 1377 1386 1395 1404 1413 1422 1431 1440 1449 1458 1467 1
476 1485 1494 1503 1512 1521 1530 1539 1548 1557 1566 1575 1584 1593 1602 161
1 1620 1629 1638 1647 1656 1665 1674 1683 1692 1701 1710 1719 1728 1737 1746
1755 1764 1773 1782 1791 1800
BUILD SUCCESSFUL (total time: 1 second)
```

图8.1 Nines程序的输出

由于NetBeans中的输出窗口不会对文本换行，所以所有的数字将在一行中显示。为了使文本能够换行，右键单击输出面板，然后从弹出的菜单中选择“Wrap Text”。

8.2 while循环

while循环不像for循环那样有多个不同的组成部分，它所需要的只是一个条件测试，由while语句来完成。下面是一个while循环语句的例子：

```
while (gameLives > 0) {
    // the statements inside the loop go here
}
```

该循环将不断重复，直到变量gameLives小于等于0。

while语句在循环一开始，即执行循环中的任何语句之前，就测试条件。因此，如果程序首次运行到**while**语句时，测试条件为**false**，循环体中的语句将根本不会执行。

如果**while**条件为**true**，将执行循环一次，然后再测试**while**条件。如果在循环体内不改变测试条件，循环将无休止地执行下去。

下面的语句使用**while**循环显示同一行文本多次：

```
int limit = 5;
int count = 1;
while (count < limit) {
    System.out.println("Pork is not a verb");
    count++;
}
```

while循环会使用在循环语句之前设置的一个或多个变量。在这个例子中，创建了两个整型变量：**limit**和**count**，其中**limit**的值为5，**count**的值为1。

该**while**循环显示文本“Pork is not a verb”4次，如果将变量**count**的初始值改为6，将不会显示这行文本。

8.3 do-while循环

do-while循环的功能类似于 **while** 循环，但测试条件的位置不同。下面是一个 **do-while**循环的例子：

```
do {  
    // the statements inside the loop go here  
} while (gameLives > 0);
```

与前面的**while**循环类似，该循环不断执行，直到变量**gameLives**不再大于0。do-while循环的不同之处在于，条件测试是在循环体语句之后而不是之前执行的。

当程序首次运行到do循环时，do和while之间的语句被自动执行，然后再测试while条件以决定是否继续循环。如果while条件为true，循环将再次执行；如果while条件为false，循环结束。在do和while语句之间，必须有改变条件的语句，否则循环将一直进行下去。do-while循环体内的语句至少会执行一次。

下列语句使用do-while循环显示相同的文本行多次：

```
int limit = 5;  
int count = 1;  
do {  
    System.out.println("I am not allergic to long division");  
    count++;  
} while (count < limit);
```

与while循环类似，do-while循环会在循环语句之前使用已经设置的一个或多个变量。

该循环显示文本“I will not allergic to long division”4次。如果将变量count的初始值设置为6，尽管count大于limit，文本仍将显示1次。

在第一次执行do-while循环时，即使循环条件为false，循环体内的语句也会被执行一次。

8.4 退出循环

退出循环的正常途径是测试条件为false，3种类型的Java循环（for、while和do-while）都是如此。然而有时可能想立即结束循环，即使此时测试条件仍然为true，为此可以使用break语句，如下面的代码所示：

```
int index = 0;
while (index <= 1000) {
    index = index + 5;
    if (index == 400) {
        break;
    }
}
```

break语句将结束包含该语句的循环体。

在本例中，只有当index变量大于1000时，while循环体才停止执行。然而，可以使用一个特殊的条件语句提前终止循环的执行：如果index等于400，将执行break语句，从而立即结束循环。

另外一个可在循环中使用的特殊语句是 **continue**，它导致退出当前循环，并进入下一次循环，请看下面的循环：

```
int index = 0;
while (index <= 1000) {
    index = index + 5;
    if (index == 400) {
        continue;
    }
    System.out.println("The index is " + index);
}
```

在该循环中，如果变量`index`的值不等于400，语句将正常执行。在本例中，`continue`语句会令循环从`while`语句开始执行，而不是执行`System.out.println()`语句。由于`continue`语句的存在，循环从来不会显示下面的文本：

```
The index is 400
```

在3种循环中，都可以使用`break`和`continue`语句。

使用`break`语句可以在程序中创建一个永远运行的循环，如下例所示：

```
while (true) {
    if (quitKeyPressed == true) {
        break;
    }
}
```

```
}
```

8.5 给循环命名

与Java程序中的其他语句类似，可将一个循环放在另一个循环中。下面的示例将一个for循环放在一个while循环中：

```
int points = 0;
int target = 100;
while (target <= 100) {
    for (int i = 0; i < target; i++) {
        if (points > 50) {
            break;
        }
        points = points + i;
    }
}
```

在这个例子中，如果变量points大于50，break语句将导致退出for循环。然而while循环永远不会结束。因为变量target永远不会大于100。

在有些情况下，你可能想退出两个循环。为此，需要给外层循环（这里是while循环）命名。要给循环命名，需要将名称放在循环起始位置的前一行，并在名称后加冒号（:）。

循环有名称后，就可以在break或continue语句中使用名称来指出它们将作用于哪个循环。下面的例子与前一个例子相同，但有一点不

同：如果变量`points`大于50，将结束两个循环：

```
int points = 0;
int target = 100;
targetLoop:
while (target <= 100) {
    for (int i = 0; i < target; i++) {
        if (points > 50) {
            break targetLoop;
        }
        points = points + i;
    }
}
```

当在`break`语或`continue`语句中使用循环的名称时，不要加上名称后面的冒号。

复杂的for循环

`for`循环可以相当复杂，可以在初始化部分、条件测试部分，以及循环体部分有多个变量。`for`循环的各个部分可以使用分号（`;`）进行隔离，而且在初始化部分可以设置多个变量，循环体中也可以有多条语句，如下面的例子所示。

```
int i, j;
for (i = 0, j = 0; i * j < 1000; i++, j += 2) {
    System.out.println(i + " * " + j + " = " + (i * j));
}
```

在for循环的每一个部分，变量之间用逗号隔开，如i=0，j=0。这个循环要显示一系列i乘以j的等式。在每次循环中，变量i加1，变量j加2。一旦i与j的乘积大于等于1000，循环将结束。

for循环的组成部分也可为空，一个这样的例子是，在程序的其他地方已经创建了计数器变量并设置了初始值，如下例所示：

```
int displayCount = 1;
int endValue = 13;
for ( ; displayCount < endValue; displayCount++) {
    // loop statements would be here
}
```

8.6 测试计算机的运行速度

本章最后一个项目是执行计算机标准检查程序（Benchmark）的Java程序，它可以测试计算机的软件或硬件的运行速度。Benchmark程序使用循环语句来重复执行下面的数学表达式：

```
double x = Math.sqrt(index);
```

该语句调用Math.sqrt()方法来查找数值的平方根。第11章将会讲解该方法的工作机制。

这里创建的Benchmark程序可以查看计算机在一分钟之内计算平方根的次数。

在NetBeans中创建一个新的Java空文件，然后命名为Benchmark。输入程序清单8.2中的文本，然后保存文件。

程序清单8.2 Benchmark.java的完整源代码

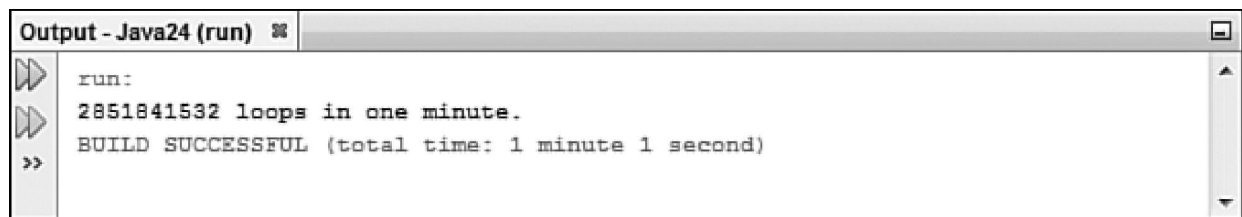
```
1: package com.java24hours;
2:
3: class Benchmark {
4:     public static void main(String[] arguments) {
5:         long startTime = System.currentTimeMillis();
6:         long endTime = startTime + 60000;
7:         long index = 0;
8:         while (true) {
9:             double x = Math.sqrt(index);
10:            long now = System.currentTimeMillis();
11:            if (now > endTime) {
12:                break;
13:            }
14:            index++;
15:        }
16:        System.out.println(index + " loops in one minute.");
17:    }
18: }
```

该程序在运行时将会进行如下操作。

- 第3~4行：声明Benchmark类，开始程序的main（）块。
- 第5行：创建变量startTime，并用当前时间（单位为毫秒）为其赋初始值。该初始值可以使用Java的System类中的currentTimeMillis（）方法来获取。

- 第6行：创建`endTime`变量，其初始值为`startTime`加60000。由于1分钟等于60000毫秒，因此`endTime`与`startTime`正好间隔1分钟。
- 第7行：创建一个长整型变量`index`，且其初始值为0。
- 第8行：`while`语句使用`true`作为测试条件，开始循环。由于测试条件始终为`true`，因此循环将一直运行（也就是说，只有其他事情将其终止时，才停止循环）。
- 第9行：计算`index`的平方根，然后将其存储到`x`变量中。
- 第10行：创建变量`now`，并使用`currentTimeMillis ()`方法将当前时间赋值给它。
- 第11~13行：如果 `now` 大于 `endTime`，这表明循环已经运行了 1 分钟，此时`break`语句会结束`while`循环。反之，继续执行循环。
- 第14行：每循环一次，`index`变量加1。
- 第16行：程序在循环体外显示它进行平方根计算的次数。

图8.2所示为该程序的输出结果。



```
Output - Java24 (run)
run:
2851841532 loops in one minute.
BUILD SUCCESSFUL (total time: 1 minute 1 second)
```

图8.2 Benchmark程序的输出

Benchmark程序非常适合用于测试读者的计算机是否比我的快。在测试时，我的计算机总共执行了29亿次计算，如果你的计算机的执行次数比这个值要高，不要只是安慰我，多买点我的书，这样我好对自己的计算机进行升级。

8.7 总结

循环是大多数编程语言的基础内容。通过循环语句对将要显示的图形依次进行控制，即可创建动画效果。如果没有循环，则在Java以及其他编程语言中都无法完成这样的任务。

8.8 问与答

问：本章在多个地方都使用了术语“初始化”，这是什么意思？

答：意思是给变量设置初始值。创建一个变量并将一个字符串赋给它时，你就是在对变量进行初始化。

问：如果循环永不结束，如何让程序停止运行？

答：在程序中，如果有循环永不结束，通常采用其他方式使其停止。例如，在游戏中，循环可能不断运行，直到玩家丧失性命为止。

编写程序时一种常见的bug是无限循环：由于编程错误，循环永不停止。如果运行Java程序时发生了无限循环错误，可以按下输出面板中左侧出现的红色警告图标。

8.9 测验

下列问题用于测试读者对循环的理解程度。为贯彻该主题的精神，不断回答这些问题，直到充分理解为止。

8.9.1 问题

1. 应使用什么分隔开for语句的各个部分?
 - a. 逗号。
 - b. 分号。
 - c. 不当班的警察。
2. 哪条语句导致程序回到循环的开头，并从那里继续运行?
 - a. conitnue。
 - b. next。
 - c. skip。
3. Java中的哪一个循环语句至少可以运行一次?
 - a. for。
 - b. while。
 - c. do-while。

8.9.2 答案

1. b. 逗号用于分隔同一部分内的内容，而分号用于分隔不同的部分。

2. a. `break`语句用于彻底结束循环，而`continue`语句用于进入下一轮循环。

3. c. 只有在执行过一次循环之后，才测试`do-while`语句中的条件。

8.10 练习

如果学习有关循环的知识后，你的头脑还清醒，请完成下面的练习以复习本章的主题：

- 修改Benchmark程序，对乘法或除法这些简单数学运算的执行进行测试。
- 使用循环编写一个小程序，找出前400个能被13整除的数。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第9章 使用数组存储信息

本章介绍如下内容：

- 创建数组；
- 设置数组的大小；
- 为数组元素赋值；
- 修改数组中的信息；
- 创建多维数组；
- 数组排序。

在计算机的发展历程之中，圣诞老人从中获得的便利要比所有人都多。几个世纪以来，人们要求他收集并处理大量的信息。年事已高的圣诞老人还必须记录下列信息：

- 淘气的孩子；
- 好孩子；
- 需要的礼物；
- 带无通路烟囱的家庭；
- 希望圣诞老人（而不是圣诞老人的老婆）满足她们愿望的妇女；
- 先对其坐骑拍照留念，然后再提需求的国家。

在北极，计算机是最实用的东西，它非常适合用于对信息进行存储、分类和研究。

在计算机程序中存储信息的最基本方式是，将它放在变量中。目前为止，你处理的所有变量只有一项，比如浮点数和字符串。

圣诞老人的好孩子名单就是一个带有许多相似信息的较大集合。你可以使用数组来记录这种名单。

数组是一组类型相关的相关变量。可以作为变量存储的任何类型的信息都能够成为存储在数组中的项目。与单个变量相比，数组可记录更复杂的信息类型，但创建和使用数组同变量一样简单。

9.1 创建数组

数组是名称相同的一组变量，读者应该很熟悉数组—想象一个销售人员展示的一系列产品，或者是具有一系列令人眼花的奖品的游戏。与变量一样，创建数组也要指出存放在数组中的变量类型以及数组名。数组与变量的不同之处在于多加了一对方括号：“[”和“]”。

像变量一样，可以创建存放任何类型信息的数组。例如，下面的语句创建一个字符串数组：

```
String[] naughtyChild;
```

下面是另外两个例子：

```
int[] reindeerWeight;  
boolean[] hostileAirTravelNations;
```

By the Way

注意

在创建数组时，Java在方括号的位置方面比较灵活，可以将方括号放在变量名后面，而不是放在变量类型的后面，如下所示：

```
String niceChild[];
```

为了使程序中的数组更容易理解，应在程序中统一使用一种格式，而不是两种格式都用。在本书使用数组的程序中，都是将方括号放在变量或对象类型的后面。

前面的例子创建了数组，但是其中没有存储任何值。为此，可以使用包含变量类型名的**new**语句，或者将初始值放在大括号“{”和“}”之间。当使用**new**关键字时，还必须指定数组包含多少项，数组中的每一项被称为一个元素。下面的语句创建了一个数组，并为其将存储的值预留了空间：

```
int[] elfSeniority = new int[250];
```

该例子创建了一个名为**elfSeniority**的整型数组，该数组包含250个元素，这些元素用于存储圣诞老人的每个精灵到北极的月份（如果圣诞老人运营一个联盟店，记录这个信息将很重要）。

使用**new**语句创建数组时，必须指定元素的个数。而且数组中的每个元素都将赋给一个初始值，初始值取决于数组的类型。对于所有数值型数组，初始值为**0**，字符型数组的初始值为**'\0'**，布尔型数组的初始值为**false**，字符串数组和所有其他对象数组的初始值为**null**。

对于不是非常大的数组，可以在创建数组时指定初始值。下面的例子创建了一个字符串数组并指定了初始值：

```
String[] reindeerNames = { "Dasher", "Dancer", "Prancer", "Vixen",  
    "Comet", "Cupid", "Donder", "Blitzen" };
```

要存储到数组元素中的信息放在“{”和“}”之间，之间用逗号隔开。数组中元素的个数也就是用逗号隔开的元素数。

数组元素是有编号的，第一个元素的编号是0，以此类推。通过引用“[”和“]”中的这个数值，也可以访问数组的特定元素。前面的语句也可以使用下述代码来实现：

```
String[] reindeerNames = new String[8];  
reindeerNames[0] = "Dasher";  
reindeerNames[1] = "Dancer";
```

```
reindeerNames[2] = "Prancer";  
reindeerNames[3] = "Vixen";  
reindeerNames[4] = "Comet";  
reindeerNames[5] = "Cupid";  
reindeerNames[6] = "Donder";  
reindeerNames[7] = "Blitzen";
```

数组中的每一个元素必须具有相同的类型。在这里，是使用一个字符串来存放驯鹿的名字。

在数组创建之后，就不能增大其空间，进而增加其他的元素。即使又想起了一种最著名的驯鹿的名称，也不能将**Rudolph**作为第9个元素加入到数组**reindeerNames**中，Java编译器不允许这样做。

9.2 使用数组

在程序中使用数组就像使用变量一样，只是需要在靠近数组名的方括号内指定元素编号。你可以在允许使用变量的任何地方使用数组元素。下面的语句使用的都是本章示例中定义的数组：

```
elfSeniority[193] += 1;  
niceChild[9428] = "Eli";  
currentNation = 413;  
if (hostileAirTravelNations[currentNation] == true) {  
    sendGiftByMail();  
}
```

数组的第1个元素的编号为0，而不是1。这意味着最大的元素编号比你想象的小1。请看下面的语句：

```
String[] topGifts = new String[10];
```

这条语句创建一个字符串数组，其元素编号为0~9，如果在程序的其他地方引用topGifts[10]，将会得到一个与ArrayIndexOutOfBoundsException相关的错误消息。

在Java程序中，异常是错误的另外一个名称。这里的异常是一个“数组越界”错误，这表示程序试图使用一个已定义边界之外的数组元素。第18章将详细讲解异常。

如果你想检测数组的上界，以避免在引用数组元素时超越上界，可以使用length变量，它与数组紧密相关。length是一个整型变量，包含数组能容纳的元素数。下面的例子创建一个数组，然后报告其长度：

```
String[] reindeerNames = { "Dasher", "Dancer", "Prancer", "Vixen",  
    "Comet", "Cupid", "Donder", "Blitzen", "Rudolph" };  
System.out.println("There are " + reindeerNames.length + "  
reindeer.");
```

在这个例子中，reindeerNames.length的值为9，也就是说可以指定的最大元素编号为8。

在Java中有两种处理文本的主要方式：字符串和字符数组。使用字符串时，一种有用的技巧是将字符串中的每个字符放在字符数组的一个元素中。为此，可使用字符串的`toCharArray()`方法，它生成一个字符数组，该数组包含的元素数与字符串长度相同。

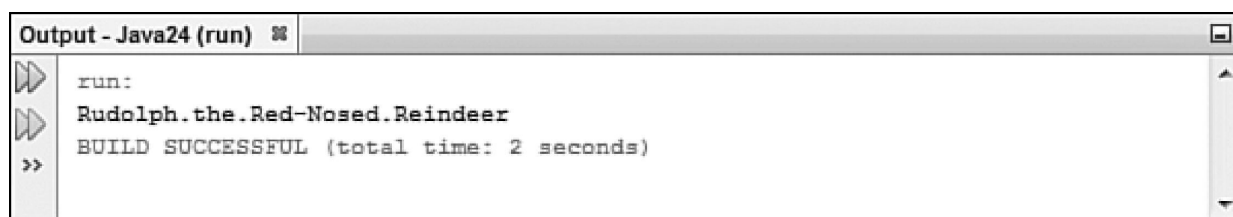
本章的第一个程序使用了本节介绍的两种方法。程序 **SpaceRemover** 将显示一个字符串，并将其中所有的空格字符（' '）替换为句点字符（'.'）。

在NetBeans中打开Java24项目，然后选择File->New File，创建一个新的Java空文件，将其命名为**SpaceRemover**。然后在源代码编辑器窗口中输入程序清单9.1中的文本，并保存。

程序清单9.1 SpaceRemover.java程序的完整文本

```
1: package com.java24hours;
2:
3: class SpaceRemover {
4:     public static void main(String[] arguments) {
5:         String mostFamous = "Rudolph the Red-Nosed Reindeer";
6:         char[] mfl = mostFamous.toCharArray();
7:         for (int dex = 0; dex < mfl.length; dex++) {
8:             char current = mfl[dex];
9:             if (current != ' ') {
10:                 System.out.print(current);
11:             } else {
12:                 System.out.print('.');
13:             }
14:         }
15:         System.out.println();
16:     }
17: }
```

在NetBeans中选择Run->Run File命令，运行该程序来查看其输出，如图9.1所示。



```
Output - Java24 (run)
run:
Rudolph.the.Red-Nosed.Reindeer
BUILD SUCCESSFUL (total time: 2 seconds)
```

图9.1 SpaceRemover程序的输出

应用程序SpaceRemover将文本“Rudolph the Red-Nosed Reindeer”存储在两个地方：一个是字符串变量mostFamous，另一个是字符数组mfl。数组是在第6行通过调用mostFamous的toCharArray()方法创建的，该方法将文本中的每个字符存储到数组的一个元素中，字符R存储在元素0中，字符u在元素1中，字符d在元素2中，依此类推，最后将字符r存储到元素29中。

第7~14行的for循环检查数组mfl中的每个字符，如果字符不是空格，就直接显示它，如果是空格，就显示句点字符（.）。

9.3 多维数组

目前为止，本章介绍的都是一维数组，使用一个数字就可以检索元素。有些类型的信息需要使用多维数组来存储，如（x, y）坐标系的点，其中数组的一维用于存储x坐标，另一维用于存储y坐标。

要创建二维数组，必须在创建和使用数组时多加一对方括号。请看下面的例子：

```
boolean[][] selectedPoint = new boolean[50][50];
selectedPoint[4][13] = true;
selectedPoint[7][6] = true;
selectedPoint[11][22] = true;
```

这个例子创建了一个名为**selectedPoint**的布尔数组，该数组的第一维有50个元素，第二维也有50个元素，因此总共有2500（50×50）个元素。该数组创建后，每个元素的默认值为**false**。接下来，将其中的3个元素设置为**true**，它们在数组中的位置分别是（4, 13）、（7, 6）和（11, 22）。

根据需要，数组可以有任意多维数，但别忘了，如果数组的维数很多，将占用大量的内存。创建50×50的**selectedPoint**数组相当于创建了2500个单独的变量。

9.4 对数组进行排序

将一系列类似的数据组织为数组后，你可以重新安排它们的序列。下面的语句交换整型数组**numbers**中两个元素的值：

```
int temporary = numbers[5];
numbers[5] = numbers[6];
numbers[6] = temporary;
```

这些语句导致number[5]和number[6]的值相互交换，整型变量temporary在交换时用作临时存储空间。“排序”是指将一组相关内容按指定顺序排列，一个例子是将数字从小到大排列。

圣诞老人可以按照姓氏对接收礼物的人进行重新排序。例如，Willie Aames和Hank Aaron会先于Dweezil Zappa和Jim Zorn收到礼物。

在Java中对数组排序很容易，因为Arrays类提供了这种功能。Arrays类位于java.util中，它可以对任何类型的数组进行排序。

要在程序中使用Arrays类，可执行下列步骤。

1. 使用import java.util.*语句将所有java.util类导入到程序中。
2. 创建数组。
3. 使用Arrays类的方法sort()来对数组重新排序。

使用Arrays类的sort ()方法对数组进行排序后，其中的值将按数字升序排列。字符和字符串将按字母顺序排列。

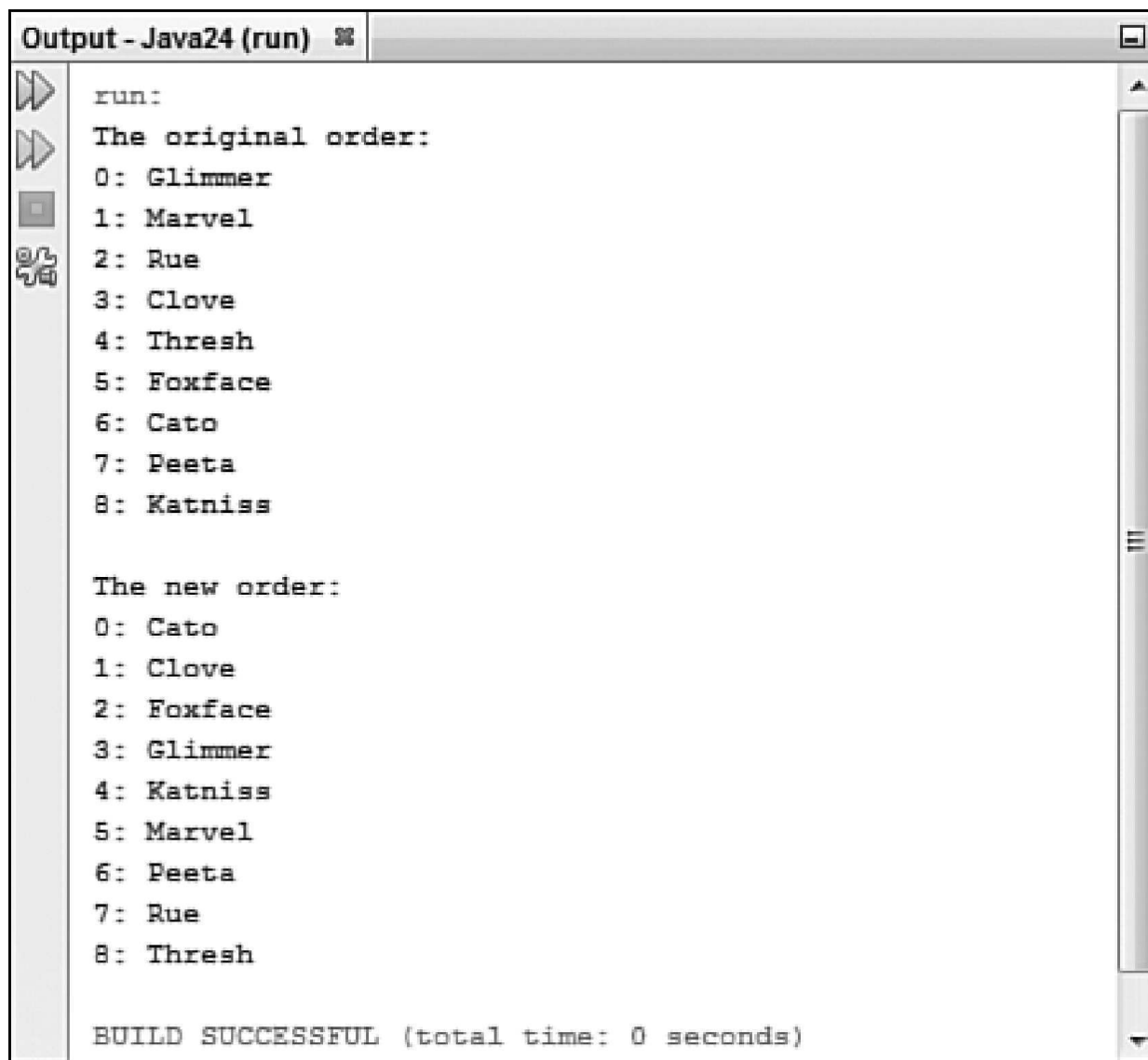
为了对其验证，创建一个新的Java空文件，将其命名为NameSorter，然后在源代码编辑器中输入程序清单9.2中的所有文本。

程序清单9.2 NameSorter.java的完整源代码

```
1: package com.java24hours;
2:
3: import java.util.*;
4:
5: class NameSorter {
6:     public static void main(String[] arguments) {
```

```
7:      String names[] = { "Glimmer", "Marvel", "Rue", "Clove",  
8:          "Thresh", "Foxface", "Cato", "Peeta", "Katniss" };  
9:      System.out.println("The original order:");  
10:     for (int i = 0; i < names.length; i++) {  
11:         System.out.println(i + ": " + names[i]);  
12:     }  
13:     System.out.println();  
14:     Arrays.sort(names);  
15:     System.out.println("The new order:");  
16:     for (int i = 0; i < names.length; i++) {  
17:         System.out.println(i + ": " + names[i]);  
18:     }  
19:     System.out.println();  
20: }  
21: }
```

当运行该Java程序时，它首先按照原来的顺序显示这9个名字，然后按照名字排序，再重新显示名字。其输出结果如图9.2所示。



```
run:
The original order:
0: Glimmer
1: Marvel
2: Rue
3: Clove
4: Thresh
5: Foxface
6: Cato
7: Peeta
8: Katniss

The new order:
0: Cato
1: Clove
2: Foxface
3: Glimmer
4: Katniss
5: Marvel
6: Peeta
7: Rue
8: Thresh

BUILD SUCCESSFUL (total time: 0 seconds)
```

图9.2 NameSorter程序的输出

如果使用字符串或基本类型（如整数和浮点数）变量时，通过Arrays类的方法只可按升序进行排序。如果要按其他顺序排列或希望排序效率比Arrays类高，可以自己编写代码来实现。

9.5 对字符串中的字符计数

在英文中，最常出现的字母依次是E、R、S、T、L、N、C、D、M和O，如果你之前看过辛迪加游戏“财富转轮”，就会意识到这一点。

By the Way

注意

如果读者不熟悉“财富转轮”这个节目，这里介绍一下：财富转轮游戏有3个参赛者，他们猜测组成一个短语、名字或引文的字母。如果猜中一个字母且是辅音字母，就可以通过转动大轮子来确定赢多少钱。为增加娱乐性，参加者与在观众席前排就坐的朋友一起来玩这个游戏，猜中字母时，就发给他们随机数量的钱，并给胜利者一个新的猜测机会。

这里将要创建的Java程序用来统计字母在不同短语或表达中出现的频率，并使用数组来存放每个字母出现的字数。当程序运行时，它将显示每个字母在短语中出现的次数。

在NetBeans中创建一个新的Java空文件，将其命名为Wheel.java，然后输入程序清单9.3中的所有文本，在输入完毕之后保存。你可以在第17行和第18行之间随意输入其他短语，只要格式与第17行相同即可。

程序清单9.3 Wheel.java的完整源代码

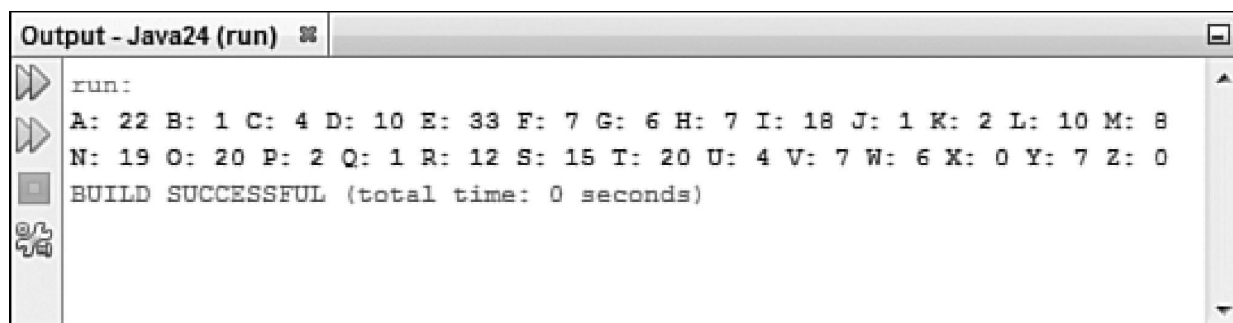
```
1: package com.java24hours;
2:
3: class Wheel {
4:     public static void main(String[] arguments) {
5:         String phrase[] = {
6:             "A STITCH IN TIME SAVES NINE",
7:             "DON'T EAT YELLOW SNOW",
8:             "JUST DO IT",
```

```

9:         "EVERY GOOD BOY DOES FINE",
10:        "I WANT MY MTV",
11:        "I LIKE IKE",
12:        "PLAY IT AGAIN, SAM",
13:        "FROSTY THE SNOWMAN",
14:        "ONE MORE FOR THE ROAD",
15:        "HOME FIELD ADVANTAGE",
16:        "VALENTINE'S DAY MASSACRE",
17:        "GROVER CLEVELAND OHIO",
18:        "SPAGHETTI WESTERN",
19:        "AQUA TEEN HUNGER FORCE",
20:        "IT'S A WONDERFUL LIFE"
21:    };
22:    int[] letterCount = new int[26];
23:    for (int count = 0; count < phrase.length; count++) {
24:        String current = phrase[count];
25:        char[] letters = current.toCharArray();
26:        for (int count2 = 0; count2 < letters.length;
count2++) {
27:            char lett = letters[count2];
28:            if ( (lett >= 'A') & (lett <= 'Z') ) {
29:                letterCount[lett - 'A']++;
30:            }
31:        }
32:    }
33:    for (char count = 'A'; count <= 'Z'; count++) {
34:        System.out.print(count + ": " +
35:            letterCount[count - 'A'] +
36:            " ");
37:        if (count == 'M') {
38:            System.out.println();
39:        }
40:    }
41:    System.out.println();
42: }
43: }

```

如果你没有添加自己的短语，该程序的输出应该如图9.3所示。



```
run:
A: 22 B: 1 C: 4 D: 10 E: 33 F: 7 G: 6 H: 7 I: 18 J: 1 K: 2 L: 10 M: 8
N: 19 O: 20 P: 2 Q: 1 R: 12 S: 15 T: 20 U: 4 V: 7 W: 6 X: 0 Y: 7 Z: 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

图9.3 Wheel程序的输出结果

Wheel程序中将会发生如下事情。

- 第5~21行：短语存储在字符串数组phrase中。
- 第22行：创建整型数组letterCount，它包含26个元素。该数组用来存储每个字母（依次为A~Z）出现的次数。元素letterCount[0]存储字母A出现的次数，元素letterCount[1]存储字母B出现的次数，依此类推，最后，元素letterCount[25]存储字母Z出现的次数。
- 第23行：开始一个for循环，该循环遍历数组phrase中的所有短语。该for语句使用了变量phrase.length，以便达到最后一个短语时结束循环。
- 第24行：创建一个名为current的字符串变量，并将数组phrase中当前元素的值赋给它。
- 第25行：创建一个字符数组，用于存储当前短语中的所有字符。
- 第26行：开始一个for循环，该循环遍历当前短语中的所有字符。这里使用了变量letters.length，以确保达到最后一个字符时结束循环。
- 第27行：创建字符变量lett并将当前字符赋给它。字符除文本值外，还有对应的数值。由于数组中的元素是带编号的，因此可以使用每个字符对应的数值来确定其元素编号。

- 第28～30行：使用if语句排除所有非字母字符，如标点符号和空格。依据当前存储在变量`lett`中的字符对应的数值，决定将数组`letterCount`中的哪个元素加1。字母对应的数值从65（代表‘A’）到90（代表‘Z’）。由于数组`letterCount`的元素编号为0～25，因此为确定将哪个数组元素加1，将变量`lett`与‘A’相减。
- 第33行：使用for循环从字母‘A’遍历到字母‘Z’。
- 第34～40行：显示当前的字母、冒号，以及该字母在数组`phrase`中存储的短语中出现的次数。如果当前的字母是‘M’，将显示一个新行，以便将输出显示在两行中。

By the Way

注意

与字符‘A’到‘Z’相关联的数值是ASCII字符集中使用的值。ASCII字符集是Unicode的一部分，后者是Java语言支持的完整字符集。Unicode字符集支持全世界的各种书面语言中使用的60000多个字符。ASCII只有256个字符。

该程序演示了如何使用两个嵌套的for循环，以每次一个字符的方式遍历一组短语。Java给每个字符提供了一个相关联的数值，这个值比数组中的字符更易使用。

9.6 总结

通过使用数组，可以在程序中存储和处理复杂的信息。数组也适合存储以列表形式存放的任何信息，并可使用第8章介绍的循环语句轻松地进行存取。

说实话，圣诞老人的信息处理需求可能超过了数组的处理能力。每年都有孩子出生，他们要求的礼物越来越复杂和昂贵。

不方便使用变量时，可以在程序中使用数组来存储信息，即使没有创建列表或，也可以使用数组。

9.7 问与答

问：字母的数值范围65（‘A’）到90（‘Z’）是基本Java语言的一部分吗？如果是，1～64留作什么用处？

答：数值1～64对应于数字、标点符号和其他不可打印的字符（如回车、换行符和退格）。与每个可打印字符相关联的数值可以在Java程序中使用，还可以使用一些不可打印的字符。Java使用Unicode编号系统，其中前127个字符来自ASCII字符集，读者在其他编程语言可能使用过。

问：为什么有些错误称为异常？

答：该术语的含义是程序正常运行时没有问题，而异常表明必须应对特殊情况。异常是Java程序发出的警告消息，在Java语言中，术语“错误（error）”有时只用于描述在运行程序的解释器中发生的错误状态。第18章将更详细地介绍这两个术语。

问：在多维数组中，可以使用变量length来测量除第一维外的其他维的长度吗？

答： 可以测量数组任何维的长度，对于第一维，可使用数组名和`length`，如`x.length`；对于其他维，可以使用该维的第1个元素和`length`。请看使用下面的语句创建的数组`data`：

```
int[][][] data = new int[12][13][14];
```

该数组第一维的长度可使用`data.length`来确定；对于第二维，可以通过`data[0].length`来测量；对于第三维，可以使用`data[0][0].length`来测量。

9.8 测验

如果读者的脑袋是数组，可以通过回答下列关于数组的问题来测量其长度。

9.8.1 问题

1. 数组最适合用于存放什么类型的信息？
 - a. 列表。
 - b. 相关的信息对。
 - c. 琐碎的东西。
2. 什么变量用于检查数组的上界？

a. top °

b. length °

c. limit °

3. 包括Rudolph在内，圣诞老人有多少只驯鹿？

a. 8 °

b. 9 °

c. 10 °

9.8.2 答案

1. a. 列表包含相同类型的信息，如字符串、数字等，适合用数组存储。

2. b. 变量length包含数组的元素个数。

3. b. Clement Clark Moore在其著作的圣诞诗歌《A Visit from St. Nicholas》中提到，圣诞老人有8只小驯鹿，所以加上Rudolph后是9只。

9.9 练习

要获得更多可供以后使用的经验，可通过下面的练习拓展有关本章主题的知识。

- 创建一个程序，它使用多维数组存储学生的成绩。第一维是学生编号，第二维是每个学生的成绩。显示全部学生的平均成绩以及每个学生的平均成绩。
- 编写一个程序，将能被13整除的前400个数存储到数组中。

要查看完成这些练习编写出的Java程序，请访问本书的配套网站 www.java.24hours.com。

第10章 创建第一个对象

本章介绍如下内容：

- 创建对象；
- 使用属性描述对象；
- 确定对象的行为；
- 合并对象；
- 从其他对象继承；
- 转换对象和其他类型的信息。

在本书中，面向对象编程（OOP）是比较难理解的专业术语之一。这个复杂的术语以优雅的方式描述了计算机程序是什么以及它是如何工作的。

在面向对象编程之前，计算机程序是使用本书前面介绍过的最简单的定义来描述的：它是文件中的一组指令，这些指令按某种可靠的顺序执行。

如果将程序视为对象的集合，便可以确定程序要完成的任务，然后将这些任务指派给最适合完成它们的对象。

10.1 面向对象编程的工作原理

可以将你创建的Java程序视为对象，就像真实世界中存在的物体一样。对象独立于其他对象而存在，以特定方式同其他对象交互，可以与其他对象合并成更大的东西。如果将计算机程序视为一组彼此交互的对象，设计出的程序将更可靠，更容易理解，更容易在其他项目中重用。

在第23章，读者将创建一个显示饼图的Java程序。饼图是一个圆，使用不同颜色的扇形区域表示数据（见图10.1）。饼图是一个由更小的对象（具有不同颜色的扇形区域、指出每个扇形区代表什么的图例以及标题）组成的对象。

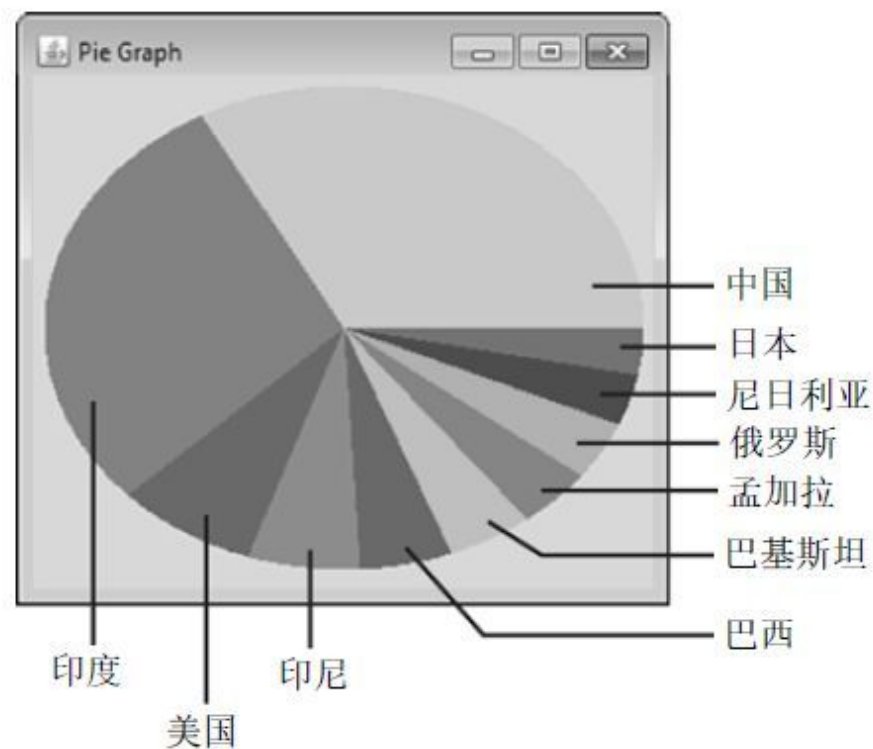


图10.1 一个显示饼图的Java程序

每个对象都有区别于其他对象的特征。饼图是圆的，而柱状图则使用一系列矩形来表示数据。如果你用划分饼图的方式划分计算机程序，也就是在进行面向对象编程（OOP）。

在面向对象编程中，对象包含两项内容：属性和行为。属性描述对象并使其不同于其他对象，而行为指的是对象能做什么。

在Java中，创建对象时使用类作为模板，“类”是对象的母版，它可以决定对象应有哪些属性和行为。读者对术语“类”应该不陌生，因为Java程序就是被称为类。使用Java创建的每个程序都是类，你可以将它用作创建新对象的模板。例如，任何使用字符串的Java程序都使用了根据String类创建的对象。String类包含属性和行为，前者决定了String对象是什么样的，而后者控制String对象能做什么。

在面向对象编程中，计算机程序是一组对象，这些对象协同工作以完成某项任务。有些简单的程序看似只有一个对象（类文件）组成，但即使是这样的程序也使用了其他对象来完成其工作。

10.2 对象示例

在显示饼图的程序中，PieChart对象可能包含以下内容：

- 计算饼图中每个扇形区应多大的行为；
- 绘制饼图的行为；
- 存储饼图标题的属性。

对读者来说，让PieChart对象来绘制它自己可能会有点奇怪，因为现实世界中图形无法绘制自身。但在面向对象编程中，对象会尽可能独立地完成工作。这种能力可以更容易使对象用于其他程序。如果PieChart对象不知道如何绘制自己，则每当在其他程序中使用PieChart对象时，你都必须创建绘制它的行为。

面向对象编程的另一个例子是自动拨号器程序。在经典的黑客电影《War Games》中，Matthew Broderick扮演的角色使用它寻找可入侵的计算机。

By the Way

注意

自动拨号器是一个软件，它使用调制解调器依次拨打一系列电话号码。这种程序旨在找到做出应答的计算机，以便以后拨打这些电话号码，看看是哪里的电话。

如果读者的年龄小于30岁，可能会很难相信计算机是使用电话进行相互连接的。要连接一台计算机，你不得不知道它的电话号码，如果它正在与另外一台计算机通话，你会得到一个忙音（表示占线）。

当前，使用自动拨号器肯定会引起本地电话公司的关注，甚至带来法律纠纷。但在1980年代，几乎不用出门就能做到。David Lightman（Broderick扮演的角色）使用自动拨号器搜索一家视频游戏公司的私有计算机系统，以便在该公司发布前就能玩其新游戏。结果Lightman找到了一台秘密的政府计算机，在该计算机上能够玩从国际象棋到《Global Therm- onuclear War》在内的所有游戏。

和其他任何计算机程序一样，自动拨号器可以被视为一组协同工作的对象。它可以分解为以下几部分。

- **Modem**对象：它具有自己的属性（比如速度）以及行为。比如，该对象能够让调制解调器拨出一个号码，并检测到另外一个计算机系统响应了该呼叫。
- **Monitor**对象：对所呼叫的号码和成功的呼叫进行记录。

每个对象都独立于其他对象存在。

设计完全独立的**Modem**对象的一个优点是，可以将其用在需要调制解调器功能的其他程序中。

使用独立对象的另一个原因是更容易调试。计算机程序的规模越来越大，如果你在调试**Modem**等对象时，你知道它不依赖于其他任何东西，则只需关注**Modem**对象完成其预期工作，并存储完成其工作所需的信息即可。

将**Java**等面向对象编程语言作为学习的第一种编程语言是有好处的，因为这样不必改正原有的编程习惯。

10.3 什么是对象

对象是通过将对象类作为模板来创建的。下面的语句将创建一个类。

```
public class Modem {  
}
```

根据这个类创建的对象毫无用处，因为它没有任何属性和行为。只有为其添加了属性和行为后，这个类才具有可用性。可以像下面这样为这个类添加属性和行为。

```
public class Modem {  
    int speed;  
  
    public void displaySpeed() {  
        System.out.println("Speed: " + speed);  
    }  
}
```

这个**Modem**类看起来应该很像本书前面编写的程序。**Modem**类以**class**语句打头，只是**class**左边还有关键字**public**。关键字**public**表示这个类是公有的，换句话说，任何程序都可以使用**Modem**对象。

Modem类的第一部分创建一个整型变量**speed**，该变量是对象的一个属性。

Modem类的第二部分是一个名为**displaySpeed()**的方法。该方法是对对象的行为的一部分，它包含一条语句：**System.out.println()**，用于显示调制解调器的速度值。

对象的变量通常称为实例变量或成员变量。

如果要在程序中使用**Modem**对象，可以使用如下语句来创建它。

```
Modem device = new Modem();
```

这条语句创建一个名为**device**的**Modem**对象。在创建了对象之后，可以设置其变量并调用其方法。要设置**device**对象的**speed**变量的值，可使用下面的语句：

```
device.speed = 28800;
```

通过调用**displaySpeed()**方法可以让这个调制解调器显示其速度，如下面的代码所示：

```
device.displaySpeed();
```

名为**device**的**Modem**对象将通过显示文本“Speed: 28800”来响应这条语句。

10.4 理解继承

OOP的一个最大优点是继承，它允许一个对象继承另外一个对象的行为和属性。

当你开始创建对象时，有时会发现将要创建的新对象和你以往开发出的对象有很多相似之处。

当《War Games》在1983年上映之时，如果David Lightman想要一个能够处理纠错且具有其他高级调制解调器特性的对象，而这些特性在当时还没有，那么他应该怎么办呢？Lightman可以通过复制Modem对象的语句然后对其修改的方式，来创建一个新的ErrorCorrectionModem对象。然而，如果ErrorCorrectionModem对象的大部分属性和行为与Modem对象相同，则上述工作完全没有必要。这也意味着如果日后要进行修改，Lightman需要升级两个独立的程序。

通过继承，程序员只需定义新类与现有类的不同之处，就能够创建一个新类。Lightman可以让ErrorCorrectionModem类继承Modem类，这样只需编写纠错调制解调器不同于以前调制解调器的部分。

要继承其他类，可使用extend语句，下面是从Modem类继承的ErrorCorrectionModem类的框架：

```
public class ErrorCorrectionModem extends Modem {  
    // program goes here  
}
```

10.5 建立继承层次

通过继承，无须做大量重复的工作就可以开发大量相关的类，它使得代码可以从一个类传递给另一个类，再传递给其他类。类的这种

组织结构称为类层次，在Java程序中使用的标准类都属于同一个类层次。

如果知道子类和超类是什么，将更容易理解类层次。从其他类继承而来的类称为子类，被继承的类称为超类。

在前面的《War Games》示例中，Modem类是ErrorCorrectionModem类的超类；而Error CorrectionModem是Modem的子类。

在层次结构中，可以从一个类派生出多个类：ISDNModem可能是Modem的另外一个子类，因为ISDN Modem包含使其不同于纠错调制解调器的属性和行为。如果ErrorCorrection Modem有子类IntenalErrorCorrectionModem，后者将继承类层次结构中位于它上面的所有类，包括ErrorCorrectionModem和Modem。这些继承关系如图10.2所示。

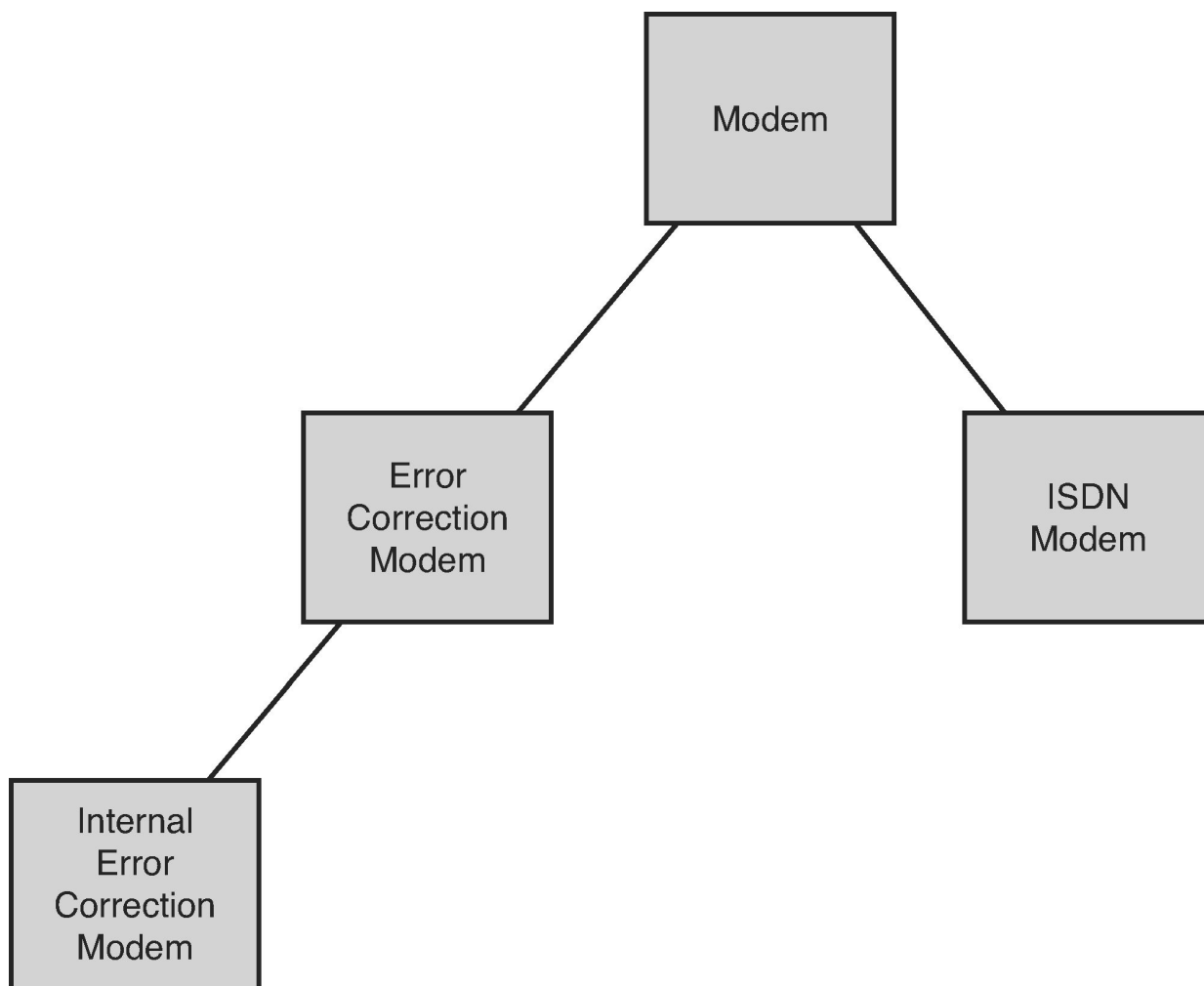


图10.2 类层次示例

组成标准Java语言的类充分利用了继承，了解这一点很有必要。第12章将详细地介绍继承。

10.6 转换对象和简单变量

在Java中需要完成的一种最常见的任务是，将信息从一种格式转换为另一种格式。可以执行多种转换，如下所示：

- 将一个对象转换为另一个对象；

- 将一个简单变量转换为另一种类型的变量；
- 使用对象创建简单变量；
- 使用简单变量创建对象。

简单变量的类型为第5章介绍的基本数据类型，包括int、float、char、long、double、byte和short。

在程序中使用方法或表达式时，必须使用方法或表达式所要求的正确的信息类型。例如，需要Calendar对象的方法必须接收Calendar对象。如果使用的方法接受单个整型参数，但传递的却是浮点数，在编译程序时将报错。

By the Way

注意

诸如System.out.println()这样的方法需要字符串参数时，你可以使用+运算符将参数中不同类型的信息合并起来。只要待合并中的信息有一个是字符串，则合并后的参数将被转换为字符串。

将信息转换为另一种类型称为类型转换。类型转换得到的信息的类型与原来的变量或对象不同。进行类型转换时，并不会修改原来的变量或对象的值，而是创建一个所需类型的新变量或对象。

讨论类型转换时，术语“源”和“目标”很有帮助。“源”指的是原始格式的信息（无论是变量还是对象），而“目标”是转换后得到的新版本。

10.6.1 简单变量的类型转换

对于简单变量，最常见的是在数值变量之间进行类型转换，如整数和浮点数之间。有一种类型的变量不能进行类型转换，这就是布尔值。

要将信息转换为一种新的类型，可以在变量前指定新类型，并将其用括号括起。例如，要将变量转换为**long**变量，可在前面加上**(long)**。下面的语句将下面的语句将**float**值转换为**int**值：

```
float source = 7.06F;  
int destination = (int) source;
```

在类型转换中，如果目标类型的取值范围比源类型大，转换将很容易，如将**byte**值转换为**int**值。**byte**的取值范围为-128~127，而**int**的取值范围为 $-2.1 \times 10^9 \sim 2.1 \times 10^9$ 。无论**byte**变量存储的什么值，新的**int**变量都有足够的空间存储它。

有时不用进行类型转换就可使用类型不同的变量，例如，**char**变量可直接用作**int**变量，**int**变量可直接用作**long**变量，而任何数值变量都可直接用作**double**变量。

在大多数情况下，由于目标类型的取值范围大于源类型，转换时不会修改信息。特例是将**int**或**long**变量转换为**float**变量，以及将**long**变量转换为**double**变量。

要从取值范围大的类型转换为取值范围小的类型，必须显式地进行转换，如下面的语句所示：

```
int xNum = 103;  
byte val = (byte) xNum;
```

这里将一个名为xNum的int变量转换为一个名为val的byte变量。在这里，目标变量的取值范围比源类型小，byte变量能够存储-128~127的整数，而int变量的取值范围大得多。

如果在转换操作中，源变量的值在目标变量中放不下，Java将修改源值，以便能够存储。如果不想改变值，这将导致意想不到的结果。

10.6.2 对象类型转换

当源对象和目标对象存在继承关系时，就可以在它们之间进行类型转换，其中一个类必须是另一个类的子类。

有些对象根本不需要转换。可在任何需要超类的地方使用类对象。Java中的所有对象都是Object类的子类，因此可在任何需要Object的地方使用任何对象。

也可以将对象用于需要其子类的地方，然而，由于子类通常比其超类包含更多的信息，因此可能缺少某些信息。如果对象没有其子类包含的方法，而程序使用了该方法，这将导致错误。

要将对象用于需要其子类的地方，必须使用类似于下面的语句对其进行显式的类型转换：

```
public void paintComponent(Graphics comp) {  
    Graphics2D comp2D = (Graphics2D) comp;  
}
```

这将Graphics对象comp转换为一个Graphics2D对象。转换过程中不会丢失任何信息，但将得到子类定义的所有方法和变量。

10.6.3 在简单变量和对象之间进行转换

不能在对象和简单变量之间进行类型转换。在Java中，每一个简单的变量类型都有很多类，比如Boolean、Byte、Character、Double、Float、Integer、Long和Short等。这些类名的首字母都是大写的，因为它们是对象，而不是简单的变量类型。

使用这些类中定义的方法，可以将变量的值作为参数来创建一个对象。下面的语句使用5309创建一个Integer对象：

```
Integer suffix = new Integer(5309);
```

用这种方式创建对象后，可以像使用其他对象那样使用该对象。当需要再次将该值作为简单变量使用时，类也有相应的方法进行转

换。例如，要通过前面的`suffix`对象得到一个`int`值，可使用下面的语句：

```
int newSuffix = suffix.intValue();
```

这条语句将变量`newSuffix`的值设置为5309，该变量的类型为`int`。最常见的在对象和变量之间进行的转换是将字符串作为数值使用。为此，可以使用`Integer`类的`parseInt()`方法，如下例所示：

```
String count = "25";  
int myCount = Integer.parseInt(count);
```

该语句将包含文本25的字符串转换为整型值25。如果字符串的值不是有效的整数，则不进行转换。

接下来将创建一个应用程序，它将命令行参数中的字符串值转换为数值。这是通过命令行从用户那里获得输入时常用的一种技巧。

在NetBeans中打开Java24项目，选择File->New File，创建一个新的Java空文件，并命名为NewRoot。在源代码编辑器中输入程序清单10.1中的内容，然后保存。

程序清单10.1 NewRoot.java的全部代码

```
1: package com.java24hours;
2:
3: class NewRoot {
4:     public static void main(String[] arguments) {
5:         int number = 100;
6:         if (arguments.length > 0) {
7:             number = Integer.parseInt(arguments[0]);
8:         }
9:         System.out.println("The square root of "
10:             + number
11:             + " is "
12:             + Math.sqrt(number)
13:         );
14:     }
15: }
```

在运行该程序之前，必须对NetBean进行配置，使其能够运行命令行参数。选择菜单命令Run->Set Project Configuration->Customize，打开Project Properties对话框。将主类命名为com.java24hours.NewRoot，并在参数字段中输入169。单击OK按钮关闭对话框。

选择Run->Run Main Project（而不是Run File）来运行该程序。该程序将显示数值169以及该数值的平方根，如图10.3所示。

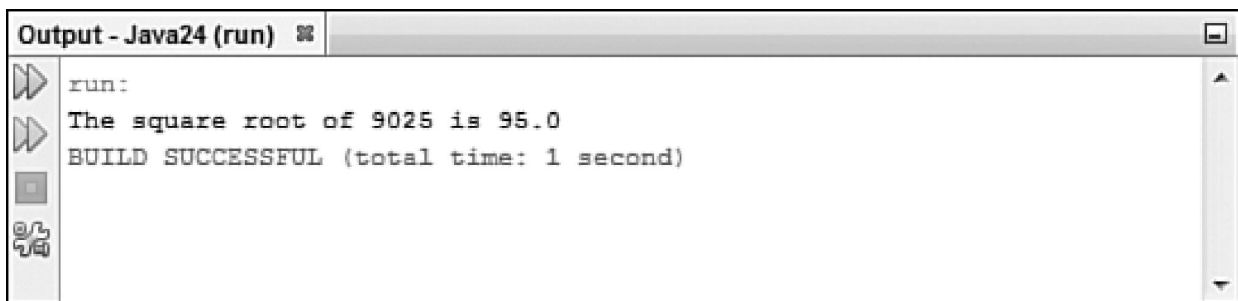


图10.3 NewRoot程序的输出

应用程序**NewRoot**是第4章示例程序的扩展，后者显示整数225的平方根。

如果该程序能够接受用户提交的数值并显示其平方根，将更有用。这就需要将字符串转换为整数。所有的命令行参数都存储在**String**数组的元素中，因此在数学表达式中使用它们之前，必须将其转换为数字。

为了根据字符串的内容创建一个整数值，可调用**Integer.parseInt()**方法，并将字符串作为唯一的参数，就像第7行那样：

```
number = Integer.parseInt(arguments[0]);
```

当程序运行时，由于 **args[0]**存储的是用户提交的第一个命令行参数，因此当以“9025”作为参数来运行该程序时，字符串“9025”被转换为整数9025。

10.6.4 自动封装和拆封

在Java中，每一种基本的数据类型都有对应的对象类：**boolean**对应**Boolean**类，**byte**对应**Byte**类，**char**对应**Character**类，**double**对应**Double**类，**float**对应**Float**类，**int**对应**Integer**类，**long**对应**Long**类，**short**对应**Short**类。

在每种基本类型及其对应的类中，可存储的值是相同的，唯一的差别是信息的格式。例如，整数值413可以用**int**变量表示，也可以用

Integer类对象表示。

Java的自动封装和拆封功能使得可以互换地使用基本数据类型及其对应的对象格式。

自动封装功能将简单变量值转换为相对应的类。

拆封功能将对象转换为相对应的简单变量值。

这些功能在幕后工作，确保在需要基本数据类型（比如**float**）时，将对象转换为该类型且值保持不变。反之亦然，即必要时自动将基本数据类型转换为相应的对象。

下面的语句显示的是自动封装和拆封是如何使用的。

```
Float total = new 1.3F;  
float sum = total / 5;
```

在 Java 的早期版本中，该代码将产生两个错误。第一条语句将一个**float**字面值1.3赋给了一个**Float**对象（这不允许），第二条语句对**Float**对象做除法，然后将值赋给一个**float**类型（这不允许）。在如今的Java版本中，这可以完美运行，因为第一条语句将字面值封装为一个对象，而第二条语句将一个对象拆封为一个原始类型。Java能够自动执行该语句，而且**sum**的结果为0.26。

在本章前面，我们看到了一个如何创建**Integer**对象的示例：


```
Integer suffix = new Integer(5309);
```

因为Java的自动封装和拆封功能，下述语句也可以完成同样的是事情：

```
Integer suffix = 5309;
```

Java意识到int值5305是存储在Integer对象中的，因此能够自动转换。

10.7 创建对象

下本章最后一个项目中，我们来看一个关于类和继承的例子，为此需要创建代表两种对象类型的类：一种是电缆调制解调器，对应的类为CableModem；另一种是DSL调制解调器，对应的类为DslModem。这里将重点放在这些对象的简单属性和行为上：

- 每个对象都有速度（speed），并可以显示出来；
- 每个对象都应能够连接到Internet。

电缆调制解调器和DSL调制解调器的一个共同点是，它们都有速度。因为这是它们共有的，可将其放在CableModem和DslModem的超

类**Modem**中。在**NetBeans**中创建一个新的**Java**空类，将其命名为**Modem**，然后在源代码编辑器中输入程序清单10.2中的文本，并保存。

程序清单10.2 **Modem.java**的完整源代码

```
1: package com.java24hours;
2:
3: public class Modem {
4:     int speed;
5:
6:     public void displaySpeed() {
7:         System.out.println("Speed: " + speed);
8:     }
9: }
```

该文件将被自动编译为**Modem.class**。该程序不能直接运行，但是可以在其他类中使用它。这个**Modem**类可以处理**CableModem**和**DslModem**类共有的行为。创建类**CableModem**和**DslModem**时，通过使用**extends**语句可以使其都成为**Modem**的子类。

在**NetBeans**中创建一个新的**Java**空文件，将其命名为**CableModem**。输入程序清单10.3中的文本，然后进行保存。

程序清单10.3 **CableModem.java**的完整源代码

```
1: package com.java24hours;
2:
3: public class CableModem extends Modem {
4:     String method = "cable connection";
5:
6:     public void connect() {
7:         System.out.println("Connecting to the Internet ...");
8:         System.out.println("Using a " + method);
9:     }
10: }
```

```
9:     }  
10: }
```

在NetBeans中创建第3个文件，将其命名为DslModem。输入程序清单10.4中的文本，然后保存。

程序清单10.4 DslModem.java的完整源代码

```
1: package com.java24hours;  
2:  
3: public class DslModem extends Modem {  
4:     String method = "DSL phone connection";  
5:  
6:     public void connect() {  
7:         System.out.println("Connecting to the Internet ...");  
8:         System.out.println("Using a " + method);  
9:     }  
10: }
```

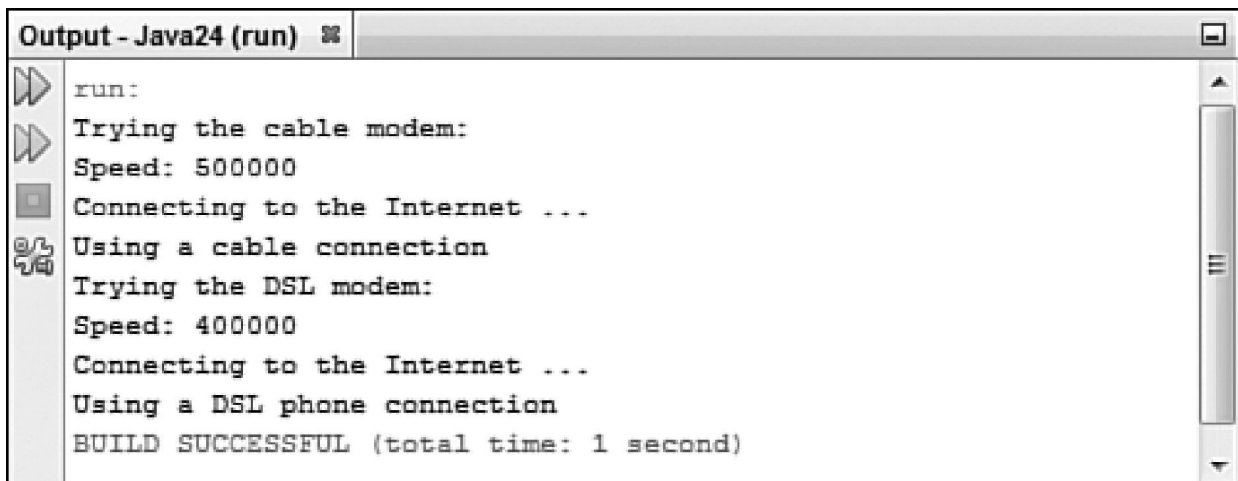
如果此时没有错误，将会得到3个类文件：Modem.class、CableModem.class和DslModem.class。然而，这3个类文件都不能运行，因为它们都没有main()块。需要创建一个小程序来测试刚才建立的类层次。

返回NetBeans，然后创建一个新的Java空文件，将其命名为ModemTester。在源代码编辑器中输入程序清单10.5中的文本，然后保存。

程序清单10.5 ModemTester.java的完整源代码

```
1: package com.java24hours;
2:
3: public class ModemTester {
4:     public static void main(String[] arguments) {
5:         CableModem surfBoard = new CableModem();
6:         DslModem gateway = new DslModem();
7:         surfBoard.speed = 500000;
8:         gateway.speed = 400000;
9:         System.out.println("Trying the cable modem:");
10:        surfBoard.displaySpeed();
11:        surfBoard.connect();
12:        System.out.println("Trying the DSL modem:");
13:        gateway.displaySpeed();
14:        gateway.connect();
15:    }
16: }
```

当运行该程序时，就会看到图10.4所示的输出。



```
Output - Java24 (run)
run:
Trying the cable modem:
Speed: 500000
Connecting to the Internet ...
Using a cable connection
Trying the DSL modem:
Speed: 400000
Connecting to the Internet ...
Using a DSL phone connection
BUILD SUCCESSFUL (total time: 1 second)
```

图10.4 ModemTester程序的输出

下面是对该程序的详细解释。

- 第5～6行：创建两个新对象，一个是名为surfBoard的CableModem对象，另一个是名为gateway的DslModem对象。
- 第7行：将CableModem对象surfBoard的speed变量设置为500000。
- 第8行：将DslModem对象gateway的speed变量设置为400000。
- 第10行：调用surfBoard对象的displaySpeed()方法，该方法是从Modem类继承的，虽然CableModem类中没有这个方法，但也可以调用它。
- 第11行：调用surfBoard对象的connect()方法。
- 第12行：调用gateway对象的displaySpeed()方法。
- 第13行：调用gateway对象的connect()方法。

10.8 总结

创建第一个对象类并建立包含几个类的层次结构后，读者应该对术语“面向对象编程”有更深入的理解。在接下来的两章中，读者将开始创建更复杂的对象，学到更多有关对象行为和属性的知识。

有了更多面向对象编程的经验后，读者将更深入地理解术语“程序”、“类”和“对象”的含义。面向对象编程需要花段时间才能习惯，但一旦掌握它，将发现它是一种设计、开发和调试计算机程序的高效方式。

10.9 问与答

问：一个类可以继承多个类吗？

答：这在有些编程语言中（比如C++）是可以的，但在Java中不行。多继承是一项功能强大的特性，但也使得面向对象编程更难于学习和使用。Java的开发者决定对继承进行限制，即任何类只能有一个超类，虽然同一个类可以有多个子类。一种补偿这种限制的方式是，可以从名为接口的这种特殊类继承方法。第19章将更详细地介绍接口。

问：什么时候应创建非public的方法？

答：当你不想让方法被其他程序使用时，就需要对该方法进行限制，使其仅用于你编写的程序。如果创建了一款游戏程序，而且你编写的shootRayGun()方法仅适合用于该游戏，可将其声明为private的。要将方法声明为private的，可省略方法名前面的public。

问：为什么可将char值当做int值来使用？

答：由于每一个字符在字符集中都有一个可以表示其位置的数值编码，因此可将字符值当做int值来使用。如果有一个变量k，其值为67，则对(char) k进行转换时，则会生成字符值'C'，原因是在ASCII字符集中，与大写字母C对应的数值编码是67。ASCII字符集是Unicode字符标准的一个子集，其中后者由Java语言采用。

10.10 测验

回答下面的问题，以测试读者对有关对象及使用对象的程序的理解程度。

10.10.1 问题

1. 什么语句让一个类能够继承另一个类？
 - a. inherits。
 - b. extends。
 - c. handItOverAndNobodyGetsHurt。
2. 为什么编译后的Java程序的文件扩展名为.class？
 - a. Java的开发者认为这是一种优等（classy）语言。
 - b. 这是对全球教师的一种巧妙赞扬。
 - c. 任何Java程序都是一个类。
3. 对象由哪两项内容组成？
 - a. 属性和行为。
 - b. 命令和数据文件。
 - c. 唾液和醋。

10.10.2 答案

1. b. 之所以使用**extends**语句，是因为子类是对超类及其超类的属性和行为的扩展。
2. c. 程序至少由一个主类以及所需的其他类组成。

3. a. 从某种意义上说，b也对，因为命令相当于行为，而数据文件相当于属性。

10.11 练习

如果读者不反对（object），可通过下面的练习拓展（extends）有关本章主题的知识。

- 创建一个AcousticModem类，其速度为300，且具有自己的connect()方法。
- 在Modem项目中，在某个类中添加一个disconnect()方法，让各种调制解调器（电缆调制解调器、DSL调制解调器、声学调制解调器）能够断开连接。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第11章 描述对象

本章介绍如下内容：

- 为类或对象创建变量；
- 使用对象和类的方法；
- 调用方法并返回一个值；
- 创建构造函数；
- 给方法传递参数；
- 使用this来引用对象；
- 创建新对象。

上一章在介绍面向对象编程时讲到，对象是一种组织程序的方式，以便拥有完成任务所需的一切。对象由属性和行为组成。

属性是存储在对象中的信息，可以是整型、字符型或布尔变量，也可以是其他对象，如String对象和Calendar对象。行为是对象中用于处理特定作业的一组语句。每组语句称为一个方法。

到目前为止，读者在无意识的情况下使用了对对象的方法和变量。如果语句中包含句点，但它不是小数点或字符串的一部分，则很可能使用了对对象。

11.1 创建变量

本章将介绍一个简单的名为**Virus**的类对象，它活着的唯一目标就是尽可能地繁殖，有点像我在大学时认识的人。为完成其工作，**Virus**需要执行多种操作，这些都被实现为类的行为。方法需要的信息将存储在属性中。

对象的属性表示对象为完成其任务所需的变量。这些变量可以为的数据类型，如整数、字符、浮点数，也可以为数组或类对象，如**String**和**Calendar**。对象的变量可以在对象所包含的任何一个方法中使用。按照惯例，在创建类的**class**语句之后和在创建方法之前，需要立即创建变量。

Virus对象需要一种方式指出文件已感染病毒。有些计算机病毒修改这样的字段，即存储文件最后修改时间的字段，例如，有些病毒可能将时间从**7:41:20**改为**7:41:61**。由于正常情况下，文件不可能在一分钟的第**61**秒修改，因此该时间表明文件感染了病毒。

Virus对象在整型变量**newSeconds**中使用了**86**这个数值（由于秒数的最大值为**60**，因此在这个数值是不正确的）。下面的语句创建一个名为**Virus**的类，它有一个名为**newSeconds**的属性以及其他两个属性：

```
public class Virus {  
    public int newSeconds = 86;  
    public String author = "Sam Snett";  
    int maxFileSize = 30000;  
}
```

这3个变量都是类的属性，它们是newSeconds、maxFileSize和author。

在变量声明语句中指定public等关键字称为访问控制，因为它决定了其他类对象将如何使用该变量或它们是否可以使用这些变量。

将变量声明为public后，使用Virus对象的其他程序也可以来修改该变量的值。

例如，如果另外一个程序为数值92赋予了特别重要的意义，则它可以将newSeconds变量的值改为92。下面的语句创建一个名为influenza的Virus对象，并设置其变量newSeconds：

```
Virus influenza = new Virus();  
influenza.newSeconds = 92;
```

在Virus类中，变量author也是public的，因此也可以在其他程序中随意修改它的值。另一个变量masFileSize只能在其所属的类中使用。

将类中的变量声明为public时，该类将无法控制其他程序使用该变量的方式。在很多情况下，这可能不是问题。例如，可将author变量该为任何表示病毒作者的名字或假名。如果制造病毒的作者被起诉，则他的名字可能会出现在法庭文件中，所以不要用一个很傻的名字。

通过限制变量的访问权限，可避免因其他程序设置错误的值而导致错误。就Virus类而言，如果newSeconds被设置为60或更小，就不能

成为报告文件是否感染病毒的可靠方式。因为有些文件无论是否感染病毒，都可能在那个秒数被保存。要避免Virus类出现这样的问题，需要做下面的两项工作：

- 将变量从public该为protected或private，后两者提供了更严格的访问限制；
- 添加修改变量值的行为以及向其他程序报告变量的值。

protected变量只能在其所在的类、该类的子类以及同一个包（package）的其他类中使用。包是一组用于完成相同目标的相关类。例如，java.util包包含很多有用的工具，如日期和时间编程以及文件归档。通过在Java程序中使用import和*，如import java.util.*，就可以在程序中很容易地引用包中的所有类。

private变量的访问限制比protected变量更严格，只能在其所属的类中使用。除非对变量做任何修改都不会影响类的功能，否则应将变量声明为private或protected。

下面的语句将变量newSeconds声明为private的：

```
private int newSeconds = 86;
```

如果要想让其他程序能够使用变量newSeconds，必须创建相应的行为，这将在本章后面介绍。

还有一种访问控制类型：创建变量时不指定`public`、`private`或`protected`。

在本章之前创建的大多数程序中，就没有指定上述任何访问控制。没有设置访问控制时，变量只能在同一个包的类中使用。这被称为默认访问控制或包访问控制。

11.2 创建类变量

创建对象时，它将拥有相应类中所有变量的版本。每个 **Virus** 对象都有自己的 `new Seconds`、`maxFileSize`和`author`变量。如果修改对象的变量，将不会影响其他**Virus**对象中的同一个变量。

有时属性与整个类而不是特定对象相关联，它们称之为类变量。如果要跟踪在程序中使用了多少个**Virus**对象，则可以使用一个类变量来存储这种信息。而且整个类只有该变量的一个拷贝。前面为对象创建的变量称为“对象变量”，因为它们与具体对象相关联。

这两种变量的创建方法和使用方法相同，但是创建类变量时指定使用关键字`static`。下面的语句为**Virus**类创建了一个类变量：

```
static int virusCount = 0;
```

修改类变量的方法与修改对象变量完全相同。如果有一个名为 `tuberculosis`的**Virus**对象，可以使用下面的语句来修改类变量

virusCount:

```
tuberculosis.virusCount++;
```

由于类变量用于整个类而不是特定对象，因此可以直接使用类名：

```
Virus.virusCount++;
```

这两条语句完成相同的工作，但处理类变量时使用类名有个优点，即表明**virusCount**是个类变量而不是对象变量。如果处理类变量时使用对象名，在不仔细查看类的源代码的情况下，将无法确定是类变量还是对象变量。

类变量也称为静态变量。

Watch **Out!**

警告：

尽管类变量很有用，但是也不要过度使用它。因为这些变量在类运行之时就会一直存在。如果类变量中存储的是一个大型的对象数组，则会占据很大的一块内存，而且不会将其释放掉。

11.3 用方法来创建行为

属性用于记录有关对象类的信息，但是要让类实现它的目的，必须创建行为。行为描述了类中完成特定任务的不同部分，每一部分都称为方法。

读者虽然没有意识到，但在前面的程序中一直在使用方法，尤其是`println()`。该方法在屏幕上显示文本。和变量一样，方法也是通过对象或类来使用的：在对象名或类名后跟句点和方法名，如`object2.move()`和`Integer.parseInt()`。

By the Way

注意

方法`System.out.println()`有点令人困惑，因为其中包含两个句点。这是因为其中涉及两个类：`System`和`PrintStream`。`System`类有一个名为`out`的变量，后者是一个`PrintStream`对象，而`println()`是`PrintStream`类的一个方法。因此，`System.out.println()`语句的意思是，使用`System`类的`out`变量的`println()`方法。可以使用这种方法串接对变量和方法的引用。

11.3.1 声明方法

创建方法的语句与创建类的语句有点类似，它们都可以在名称后面的括号中指定参数，都使用大括号“{”和“}”指示开始和结束。不同之处在于，方法可以在执行完毕后返回一个值，返回值可以是简单类型（如整数或布尔值），也可以是对象。

下面是一个**Virus**类可用于感染文件的方法：

```
public boolean infectFile(String filename) {  
    boolean success = false;  
    // file-infecting statements go here  
    return success;  
}
```

该方法接受一个参数—名为**filename**的字符串变量，该变量表示要攻击的文件。如果感染成功，**success**变量的值将被设置为**true**。

在方法开头的语句中，方法名**infectFile**前有**boolean**。该关键字指出方法执行完毕后将返回一个布尔值。实际用于返回值的是**return**语句，这里返回的是**success**的值。

如果方法不应该返回值，则方法名前面使用**void**关键字。

当方法返回一个值时，可以将方法用于表达式中。例如，如果创建了**Virus**对象**malaria**，可以使用下面这样的语句：

```
if (malaria.infectFile(currentFile)) {  
    System.out.println(currentFile + " has been infected!");  
} else {  
    System.out.println("Curses! Foiled again!");  
}
```


当方法返回值时，这个方法可用于程序中任何可以使用变量的地方。

在本章前面，为防止其他程序读取或修改变量newSeconds，将其声明为private。

也可以通过其他方式在别处使用newSeconds变量：在Virus类中创建读写newSeconds变量的public方法。不同于变量newSeconds本身，这些新方法应该是public的，这样其他程序才能调用它们。

请看下面两个方法：

```
public int getSeconds() {  
    return newSeconds;  
}  
  
public void setSeconds(int newValue) {  
    if (newValue > 60) {  
        newSeconds = newValue;  
    }  
}
```

这些方法称为accessor方法，原因是它们可以允许newSeconds变量被其他对象访问。

getSeconds()方法用于返回newSeconds变量的当前值，它没有任何参数，但是在方法名后面还需要有一对括号。setSeconds()方法接受一个参数：整型变量newValue。该参数是newSeconds的新值。如果newValue大于60，则进行修改。

在这个例子中，**Virus**类控制**newSeconds**变量如何被其他类使用。这个过程称为“封装”，它是面向对象编程的基本概念。对象防止自己被误用的能力越强，在其他程序中使用它时越有用。

尽管 **newSeconds** 变量是 **private**，但是新方法 **getSeconds()**和 **setSeconds()**能够处理**newSecond**变量，原因是它们位于同一个类中。

11.3.2 参数不同的类似方法

正如读者在**setSeconds()**方法中看到的，可以向方法传递参数来影响其行为。在类中，不同的方法有不同的名称，但如果接受不同的参数，方法也可以同名。

如果两个方法接受的参数数量不同或参数类型不同，它们可以同名。例如，对于**Virus**对象，有两个**tauntUser()**方法可能会很有用。其中一个方法不接受任何参数，它发出通用的嘲笑（**taunt**），另一个接受指定嘲笑作为字符串的参数。下面的语句实现了这两种方法：

```
void tauntUser() {  
    System.out.println("That has gotta hurt!");  
}  
  
void tauntUser(String taunt) {  
    System.out.println(taunt);  
}
```

这两个方法具有相同的名字，但是参数不同：一个方法没有参数，另外一个方法有一个**String**参数。传递给方法的参数称为方法签名

（signature）。在每一个方法具有不同签名的前提下，类可以具有多个名字相同的不同方法。

11.3.3 构造函数

在程序中创建对象时，使用关键字new，如下所示：

```
Virus typhoid = new Virus();
```

这条语句创建一个名为typhoid的Virus对象。使用关键字new时，将调用类的一个特殊方法，该方法称为构造函数（constructor），因为它处理创建对象所需做的工作。构造函数用于设置对象正常工作所需的变量及方法。

定义构造函数的方式与其他方法类似，只是它不能像其他方法那样返回值。下面是Virus类的两个构造函数：

```
public Virus() {  
    author = "Ignoto"; // author is a string  
    maxFileSize = 30000; // maxFileSize is an int  
}  
  
public Virus(String name, int size) {  
    author = name;  
    maxFileSize = size;  
}
```

与其他方法类似，通过使用不同的参数，可以在同一个类中定义多个构造函数。在这里，像下面那样使用关键字**new**来创建对象时，将调用第一个构造函数：

```
Virus mumps = new Virus();
```

仅当在**new**语句传递一个字符串参数和一个整型参数时，才会调用另一个构造函数，如下所示：

```
Virus rubella = new Virus("April Mayhem", 60000);
```

如果在类中不包含任何构造函数，将从超类那里继承一个不接受参数的构造函数。也可能继承其他构造函数，这取决于使用的超类。

使用**new**语句创建对象时，类中必须有一个这样的构造函数，即其参数数量和类型与**new**语句中指定的完全相同。在**Virus**类中，有两个构造函数：**Virus()**和**Virus(String name, int size)**，所以只能使用两种类型的**new**语句创建**Virus**对象：一种没有参数，另一种将一个字符串和一个整数作为参数。

**Watch
Out!**

警告：

如果超类中定义了一个或多个参数的构造函数，则类不再从其超类中继承无参数的构造函数。出于这个原因，当你的类有其他构造函数时，你必须总是定义一个无参数的构造函数。

11.3.4 类方法

与类变量相似，类方法提供与整个类而不是特定对象相关联的方法。当方法不影响单个对象时，将其声明为类方法。一个这样的例子是前一章使用的Integer类的parseInt()方法，该方法用于将字符串转换为整型变量，如下所示：

```
int fontSize = Integer.parseInt(fontText);
```

这是一个类方法。要将方法声明为类方法，在方法名前使用关键字static，如下所示：

```
static void showVirusCount() {  
    System.out.println("There are " + virusCount + " viruses");  
}
```

在本章前面，使用类变量 virusCount 来记录程序创建了多少个Virus对象。方法 showVirusCount()是一个类方法，它显示创建的对象总数，可使用下面这样的语句来调用它：

```
Virus.showVirusCount();
```

11.3.5 方法中变量的作用域

当在类的方法中创建变量或对象时，它们只在该方法内可用，原因在于变量的作用域。作用域是程序中变量所在的语句块。如果离开了作用域定义的程序部分，就不能使用该变量。

程序中的大括号“{”和“}”定义变量的作用域。在大括号内创建的任何变量不能在大括号外使用，例如，请看下面的语句：

```
if (numFiles < 1) {  
    String warning = "No files remaining.";  
}  
System.out.println(warning);
```

这段代码不能正常运行（在NetBeans中不能被编译），因为warning变量是在if块句的大括号之间创建的，这对括号定义了该变量的作用域。在这对大括号外，变量warning不存在，因此System.out.println()方法不能将它用作参数。

在一对大括号内又使用另一对大括号时，要注意使用的变量的作用域，请看下面的例子：

```
if (infectedFiles < 5) {
```

```
int status = 1;
if (infectedFiles < 1) {
    boolean firstVirus = true;
    status = 0;
} else {
    firstVirus = false;
}
}
```

看到问题了么？在这个例子中，`status`变量可以在任何地方使用，但是为`firstVirus`变量赋值的语句将导致编译错误，因为`firstVirus`变量是在`if (infectedFiles < 1)`语句内创建的，在后面的`else`语句中不存在。

要修复这种问题，必须在这两个语句块外创建`firstVirus`变量，这样其作用域将包含这两个语句块。一种解决办法是在创建`status`变量之后创建`firstVirus`变量。

作用域规则使程序更容易调试，因为作用域限制了变量的使用区域。这避免了在其他编程语言中最常见的一种错误：在程序的不同部分以不同的方式使用相同的变量。

作用域的概念也适用于方法，因为它们是用左大括号（`{`）和右大括号（`}`）定义的。在一个方法中创建的变量不能在其他方法中使用。如果变量是作为对象变量或类变量创建的，才可以在多个方法中使用。

11.4 将一个类放在另一个类中

虽然Java程序也称为类，但很多时候一个程序需要多个类才能完成其工作。包含多个类的程序有一个主类和任意数目的辅助类组成。

将程序分成多个类时，有两种方法可用于定义辅助类。一种是分别定义每个类，如下例所示：

```
public class Wrecker {
    String author = "Ignoto";

    public void infectFile() {
        VirusCode vic = new VirusCode(1024);
    }
}

class VirusCode {
    int vSize;

    VirusCode(int size) {
        vSize = size;
    }
}
```

在本例中，VirusCode类被用作Wrecker类的辅助类。通常在同一个源代码文件中定义辅助类，因为它们用于辅助目的。源文件编译后，将生成多个类文件。上面的例子在编译时将生成文件Wrecker.class和VirusCode.class。

Watch Out!

警告：

如果在同一个源文件中定义了多个类，只能有一个类为**public**，其他类在它们的类语句中不能指定为**public**。另外，源代码文件的名称应与它定义

的public类的名称匹配。

创建一个主类和一个辅助类时，也可以将辅助类放在主类中。在这种情况下，辅助类称为内部类。

内部类放在另一个类的左大括号和右大括号之间。

```
public class Wrecker {
    String author = "Ignoto";

    public void infectFile() {
        VirusCode vic = new VirusCode(1024);
    }

    class VirusCode {
        int vSize;

        VirusCode(int size) {
            vSize = size;
        }
    }
}
```

内部类的使用方法与其他辅助类相同，主要差别（除位置外）出现在编译之后。内部类没有使用class语句来指定名称，而是由编译器给它们指定名称，而且名称中包含了主类的名称。

在上一个示例中，编译器将生成Wrecker.class和Wrecker\$VirusCode.class。

11.5 使用关键字this

由于可以同时引用其他类的变量和方法以及当前类的变量和方法，所以有时候引用的变量将不清楚。为了使变量引用更清晰，一种解决办法是使用关键字**this**，它引用程序本身的对象。

使用对象的方法或变量时，将对象名放在变量名或方法名之前，并用句点隔开，请看下面的例子：

```
Virus chickenpox = new Virus();
chickenpox.author = "LoveHandles";
chickenpox.setSeconds(75);
```

这些语句创建了一个名为chickenpox的Virus对象，设置chickenpox的name变量，然后调用chickenpox的setSeconds()方法。

在程序中有时需要引用当前对象，即程序本身代表的对象。例如，在Virus类中，可能有一个方法，该方法有自己的author变量：

```
public void checkAuthor() {
    String author = null;
}
```

在这个例子中，chickAuthor()方法的作用域内存在一个名为author的变量，它不同于对象的变量author。如果要引用当前对象的author变量，必须使用关键字**this**，如下所示：

```
System.out.println(this.author);
```

通过使用**this**，可以明确地指出要引用的变量或方法。在类中，可以**this**代替对象名。例如，如果要将当前对象作为一个参数传递给方法，可以使用下面的语句：

```
verifyData(this);
```

在很多情况下，无需使用**this**就能清晰地引用对象的变量或方法。然而，使用**this**确保引用明确也没有什么害处。

当设置对象的变量值时，这个**this**关键字将在构造函数中发挥作用。来看具有**author**和**maxFileSize**变量的**Virus**对象，下面这个构造函数将会设置这两个变量：

```
public Virus(String author, int maxFileSize) {  
    this.author = author;  
    this.maxFileSize = maxFileSize;  
}
```

11.6 使用类方法和类变量

在我们的律师的坚持下，本章的最后一个项目将不再创建病毒程序，而创建一个简单的Virus对象，它可以统计程序创建的Virus对象的数量并报告该数量。

在NetBeans选择File->New File，创建一个新的Java空文件，将其命名为Virus，然后在源代码编辑器中输入程序清单11.1中的内容，并保存。

程序清单11.1 Virus.java程序的完整版本

```
1: package com.java24hours;
2:
3: public class Virus {
4:     static int virusCount = 0;
5:
6:     public Virus() {
7:         virusCount++;
8:     }
9:
10:    static int getVirusCount() {
11:        return virusCount;
12:    }
13: }
```

当文件保存后，NetBeans将自动编译。这个类缺乏main()方法，因此不能直接运行。为了测试这个新的Virus类，需要再创建一个能创建Virus对象的类。

VirusLab类是一个简单的应用程序，它可以创建Virus对象，然后通过Virus类的getVirusCount()方法来计算创建的对象数量。

在NetBeans中新建一个文件，输入程序清单 11.2 所示的内容，完成后将文件保存为VirusLab.java。

程序清单11.2 VirusLab.java程序的完整版本

```
1: package com.java24hours;
2:
3: public class VirusLab {
4:     public static void main(String[] arguments) {
5:         int numViruses = Integer.parseInt(arguments[0]);
6:         if (numViruses > 0) {
7:             Virus[] virii = new Virus[numViruses];
8:             for (int i = 0; i < numViruses; i++) {
9:                 virii[i] = new Virus();
10:            }
11:            System.out.println("There are " +
Virus.getVirusCount()
12:                + " viruses.");
13:        }
14:    }
15: }
```

VirusLab类是一个应用程序，运行时从命令行接受一个参数：要创建的Virus对象数。要在NetBeans中指定命令行参数，请执行如下操作。

1. 选择Run->Set Project Configuration->Customize，打开The Project Properties对话框。

2. 在Main Class 字段输入“VirusLab”，在Arguments字段输入希望程序创建的Virus对象的数量。

3. 单击OK按钮关闭对话框。

以这种方式配置过程序之后，在NetBeans中选择Run->Run Main Project命令来运行该程序。

参数将使用发送给main()方法的字符数组读入到应用程序中。在VirusLab类中，这将在第4行执行。

为了将参数用作整数，必须将其从String对象转换为整数，这就需要Integer类的parseInt()类方法。在第5行，使用通过命令行传递给程序的第一个参数创建了一个名为 num Viruses的变量。

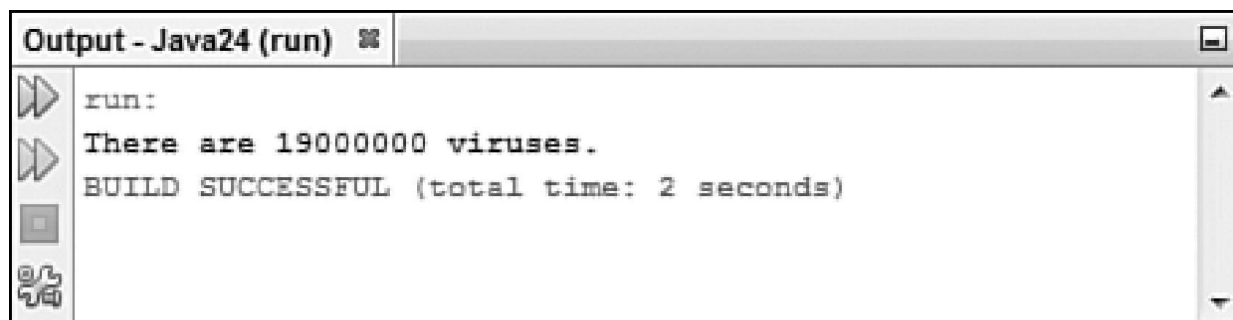
如果numViruses变量大于0，在VirusLab应用程序中将发生下列事情。

- 第7行：创建一个Virus对象数组，该数组的长度由numViruses变量指定。
- 第8~10行：用一个for循环为数组中的每个Virus对象调用构造函数。
- 第11~13行：构造完所有Virus对象后，使用Virus类的getVirusCount()方法计算创建了多少个对象。这应该与运行应用程序VirusLab时在命令行输入的参数相同。

如果numViruses变量不大于0，应用程序VirusLab将什么都不做。

编译文件VirusLab.java后，使用任何命令行参数测试该应用程序。可创建的Virus对象数量与取决于运行VirusLab应用程序的计算机系统的可用内存量。在作者的计算机系统上，当超过1亿2千8百万Virus对象时，将导致程序显示消息“OutOfMemoryError”，然后崩溃。

如果指定的Virus对象数量小于你的系统可以处理的数量，则输出应该会如图11.1所示。



```
run:
There are 19000000 viruses.
BUILD SUCCESSFUL (total time: 2 seconds)
```

图11.1 VirusLab程序的输出

11.7 总结

至此，读者已经学完了本书三章面向对象编程内容中的两章。你学习了如何创建对象、为对象和类定义行为和属性，以及如何使用类型转换将对象和变量转换为其他类型。

以对象的方式思考是学习Java编程语言面临的严峻挑战之一。然而，理解对象后，将意识到整个Java语言都是充分使用了对象和类。

下一章将介绍如何为对象指定父对象和子对象。

11.8 问与答

问： 为了使用类变量或类方法，必须创建一个对象吗？

答： 由于类变量和类方法不与特定对象相关联，不需要仅为使用它们而创建对象。 `Integer.parseInt()` 方法的使用就是一个例子，你

不需要为将字符串转换为整数而创建一个新的Integer对象。

问：有没有Java支持的所有内置方法的列表？

答： Oracle为Java语言中的所有类提供了完整的文档，包括你可以使用的所有公有方法。该文档位于
<http://download.java.net/jdk8/docs/api>。

问：有没有一种方法可以控制Java程序使用的内存量？

答： 在Java虚拟机运行一个应用程序时，可用的内存由两个因素来控制：计算机上可用的总物理内存以及为JVM配置的内存量。为JVM分配的内存默认为256MB。可以使用-Xmx命令行参数来设置一个不同的值。

要在NetBeans设置该值，选择Run->Set Project Configuration->Customize。这将打开Project Properties对话框，而且Run设置会出现在该对话框前面。在VM Options字段，输入-Xmx1024M，为JVM分配1024MB内存。可以调整该值，以分配更多或更少的内存。还要填写Main Class和Arguments字段，然后选择Run->Run Project。

11.9 测验

回答下面的问题，以检测你是否理解面向对象编程技术所需的属性和行为。

11.9.1 问题

1. 在Java类中，方法是一种什么？
 - a. 属性。
 - b. 语句。
 - c. 行为。
2. 要将变量声明为类变量，创建它时必须使用什么关键字？
 - a. new。
 - b. public。
 - c. static。
3. 变量可用的程序部分称为什么？
 - a. 窝。
 - b. 作用域。
 - c. 变量流域。

11.9.2 答案

1. c. 方法是由语句组成的，但它是一种行为。
2. c. 如果不指定关键字static，则变量为对象变量而不是类变量。

3. b. 在变量的作用域外使用变量时，编辑器将会提示错误。

11.10 练习

如果谈论病毒没有让你生病，可通过下面的练习加深对本章主题的理解。

- 在Virus类添加一个private变量，它用于存储名为newSeconds的整数。创建两个方法，其中一个返回变量newSeconds的值；另一个在提供的新值位于60和100之间时，将变量newSeconds设置为提供的值。
- 编写一个Java应用程序，它接受一个字符串参数，并依次将其转换为float变量、Float对象和int变量。使用不同的参数运行程序多次，看看结果如何。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第12章 充分利用现有对象

本章介绍如下内容：

- 超类和子类的设计；
- 建立继承层次；
- 覆盖方法。

Java对象非常适合繁殖。当创建了一个对象（一组属性和行为）时，实际上你设计了可遗传给后代的品质。这些子对象将继承父对象的许多相同的属性和行为，而且它们还可以做不同于父对象的事情。

这种系统称为继承，也就是超类（父母）遗传给子类（孩子）。继承是面向对象编程最有用的方面，本章将详细介绍它。

面向对象编程的另一个有用的方面是能够创建可在不同程序中使用的对象。可重用性使得开发没有错误的可靠程序更加容易。

12.1 继承的威力

每当读者使用标准Java类（如String或Integer）时，你使用的就是继承。Java类被组织成金字塔型的类层次结构，其中所有的类都是从Object类派生而来的。

类继承其上面所有的超类。要了解其中的原理，来看看JApplet类。这个类是所有applet的超类，后者是使用图形用户界面框架（称之

为Swing) 而且基于浏览器的Java程序。JApplet是Applet的子类。

图12.1显示了JApplet的部分家谱, 其中每个方块都是一个类, 超类与其下面的子类用直线连接。

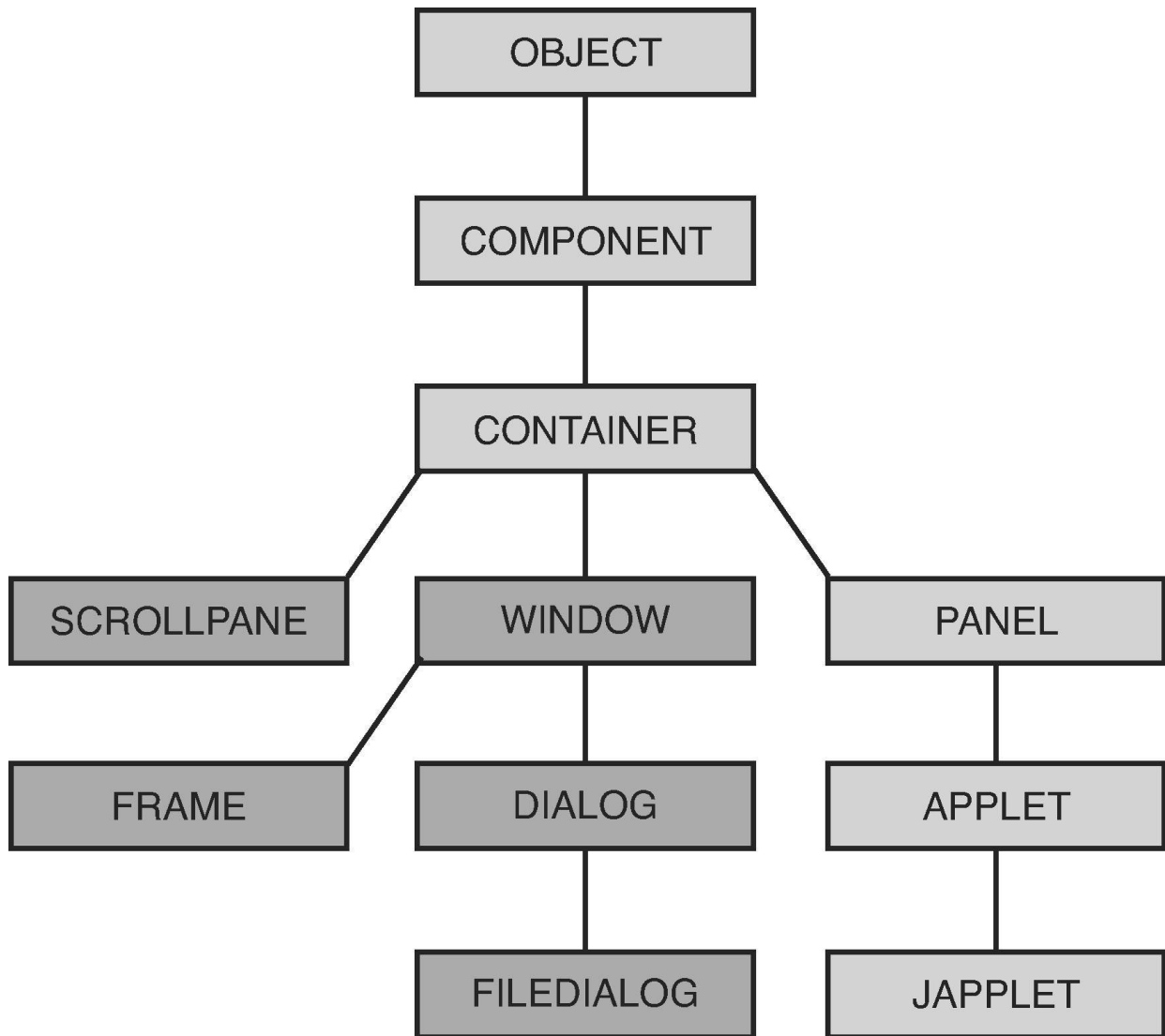


图12.1 Applet类的家谱

最顶端是Object类。在继承层次结构中, JApplet有5个超类: Applet、Panel、Container、Component、Object。

JApplet从这些类继承属性和行为，因为在超类继承层次结构中，它们都在JApplet的正上方。JApplet不继承图12.1中用阴影标识的5个类，包括Dialog和Frame，因为在继承层次结构中，它们不在JApplet的上方。

如果感到迷惑，可将层次结构视为家谱。JApplet继承父亲、祖父，不断向上直到太祖父Object类。然而，JApplet类不会从兄弟或堂兄那里继承。

创建一个新的类可以归结为执行如下任务：你需要定义其不同于现有类的地方即可，其他工作都已经为你做好。

12.1.1 继承行为和属性

类的行为和属性由两部分组成：自己的行为 and 属性以及从超类继承的行为和属性。

下面是Applet的一些行为和属性：

- equals()方法确定JApplet对象的值是否与另一个对象相同；
- setBackground()方法设置applet窗口的背景颜色；
- add()方法给applet添加用户界面组件，如按钮和文本框等；
- setLayout()方法定义如何组织applet的图形用户界面。

JApplet类可以使用所有这些方法，即使setLayout()方法不是从其他类继承来的。equals()方法是在Object类中定义的，setBackground()方法来自Component类，而add()方法来自Container类。

12.1.2 覆盖方法

在JApplet类中定义的有些方法也在其超类中定义了，例如，update()方法在JApplet和Component类中都有定义。当方法在子类和超类中都定义了时，将使用子类中的定义；因此子类可以修改、替换或完全删除超类的行为和属性。就update()方法而言，它旨在删除超类中的一些行为。

在子类中创建新方法以修改从超类继承来的行为被称为“覆盖”（overriding）方法。如果继承的行为不能产生所需的结果，则需要覆盖相应的方法。

12.2 建立继承

使用关键字extends将一个类指定为另一个类的子类，如下所示：

```
class AnimatedLogo extends JApplet {  
    // behavior and attributes go here  
}
```

上述语句创建AnimatedLogo类，并将其指定为JApplet的子类。所有的Swing applet都必须是JApplet的子类，因为在Web页面中显示时，它们需要这个类提供的功能才能运行。

AnimatedLogo必须覆盖方法paint()，该方法用于绘制要在程序窗口中显示的所有内容。paint()方法是由Component类实现的，并向下传递给AnimatedLogo类，但paint()不做任何事情，Component类提供该方法旨在让其子类需要显示内容时可使用它。

要覆盖一个方法，则在声明该方法时采用的方式应该与它在其超类中的声明方式相同，即public方法仍必须是public的，方法的返回值类型必须相同，且参数的数量和类型都不能变。

在Component类中，paint()方法的声明如下：

```
public void paint(Graphics g) {
```

在AnimatedLogo中覆盖该方法时，必须使用类似于下面的语句开始：

```
public void paint(Graphics screen) {
```

唯一的差别是Graphics对象的名称，这对于判断创建方法的方式是否相同没有影响。这两个语句是相同的，因为下面几项相同：

- 两个paint()方法都是public的；
- 两个方法都没有返回值，因为声明时使用了关键字void；
- 都接受Graphics对象作为唯一的参数。

在子类中使用this和super

在子类中this和super是两个很有用的关键字。

前一章讲到，关键字**this**用于引用当前对象。创建类时，要引用根据该类创建的特定对象，可使用**this**，如下面的语句所示：

```
this.title = "Cagney";
```

这条语句将对象的**title**变量设置为文本**Cagney**。

关键字**super**的用途与此类似：引用对象的上一级超类。可以下面几种方式使用**super**：

- 引用超类的构造函数，如**super("Adam", 12);**
- 引用超类的变量，如**super.hawaii = 50;**
- 引用超类的方法，如**super.dragnet()**。

使用关键字**super**的方式之一是在子类的构造函数中。子类从其超类继承行为和属性，因此必须将子类的构造函数与超类的构造函数关联起来，否则有些行为和属性可能无法正确设置，导致子类不能正常工作。

为了关联构造函数，在子类的构造函数中，第一条语句必须调用超类的构造函数，因此需要使用关键字**super**，如下面的语句所示：

```
public DataReader(String name, int length) {  
    super(name, length);  
}
```

这是一个子类构造函数的例子，它使用`super(name, length)`调用超类相应的构造函数。

如果不使用`super()`来调用超类的构造函数，则在子类构造函数执行时，**Java**将自动调用无参数的超类构造函数。如果该超类构造函数不存在或提供了意料之外的行为，将导致错误，因此最好手工调用超类的构造函数。

12.3 使用现有的对象

面向对象编程鼓励代码重用。如果在某个**Java**编程项目中开发了一个对象，可以将其用于其他项目中，而无需做任何修改。

如果**Java**类是设计良好的，完全可以在其他程序中使用它。在程序中可以使用的对象越多，编写软件需要做的工作就越少。如果有优秀的拼写检查对象能够满足你的需要，便可以使用它而不用自己编写。这甚至可以给老板错觉：将拼写检查功能添加到程序中需要做很多的工作，这样你就有更多的时间在办公室打长途电话。

By the Way

注意

本书作者像很多人一样，是个自由职业者，在家办公。评估其上述建议时，别忘了考虑工作环境。

Java刚面世时，共享对象的系统很不正规。程序员尽可能独立地开发其对象，并且通过使用私有变量和读写这些变量的公有方法来进行保护，以防误用。

有了开发可重用对象的标准后，共享对象变得功能强大起来。

标准的好处体现在下面几个方面：

- 可以更少地编写关于对象工作原理的文档，因为知道标准的人对其工作原理很清楚；
- 可以根据标准来设计开发工具，使得使用对象更容易；
- 遵循标准的两个对象可以相互交互，而无需通过特殊编程使其互相兼容。

12.4 将相同类的对象存储到数组列表中

编写计算机程序时，需要做出的一个重要决策是，将数据存储在哪儿。在本书的前半部分，介绍了3种存储信息的地方：

- 基本数据类型（如int和char）；
- 数组；
- String对象。

还有更多的地方可以存储信息，因为任何Java类都可以存储数据。其中一个最有用的类是ArrayList，这是一个存储相同类对象的数据结构。

如类名所示，数组列表与数组类似，它也可以存储相关数据的元素，但是它的大小可以随时调整。

`ArrayList`类位于`java.util`类包中，这是Java类库中最有用的一个包。在程序中使用它，可使用一条`import`语句：

```
import java.util.ArrayList;
```

数组列表存储的对象要么属于同一个类，要么有相同的超类。要创建数组列表，需要引用两个类：`ArrayList`类和列表要存储的类。

将要在列表中存储的类的名称放在“<”和“>”之间，如下述语句所示：

```
ArrayList<String> structure = new ArrayList<String>();
```

上述语句创建一个存储字符串的列表。以这种方式识别一个列表的类时，用到了泛型（**generics**），泛型用来指示数据结构（比如数组列表）可以存储的对象类型。如果你在以前的Java版本中使用列表，需要调用构造函数，如下所示：

```
ArrayList structure = new ArrayList();
```

虽然仍可以这样做，但泛型使代码更可靠，因为它向编译器提供了一种防止产生更多错误的方法。在这里，泛型通过将错误的对象类放到数组列表中，来阻止你滥用数组列表。如果试图将一个**Integer**对象存储到本应存储**String**对象的列表中，编译器将报错。

与数组不同，数组列表存储的元素数量并非固定的。列表在创建时包含**10**个元素，如果你知道需要存储更多的对象，可通过构造函数的参数指定长度。下面的语句创建一个包含**300**个元素的列表：

```
ArrayList<String> structure = new ArrayList<String>(300);
```

要将对象加入到列表中，可调用列表的**add()**方法，并将对象作为其唯一的参数：

```
structure.add("Vance");  
structure.add("Vernon");  
structure.add("Velma");
```

对象按照顺序加入到列表中，因此如果这是加入到**structure**中的前**3**个元素，则元素**0**为“**Vance**”，元素**1**为“**Vernon**”，元素**2**为“**Velma**”。

要从列表中检索元素，可使用方法`get()`并将元素的索引号作为其参数：

```
String name = structure.get(1);
```

该语句将“Vernon”存储到字符串变量`name`中。

要查看列表的元素中是否包含某个对象，可调用`contains()`方法并将该对象作为参数：

```
if (structure.contains("Velma")) {  
    System.out.println("Velma found");  
}
```

要将对象从列表中删除，可调用`remove()`方法并将该对象或其索引号作为参数：

```
structure.remove(0);  
structure.remove("Vernon");
```

这两条语句导致列表中只留下“Velma”。

遍历数组列表

Java包含了一个特殊的for循环以方便载入一个数组列表以及依次检查其每个元素。

该循环由两部分组成，比第8章介绍的for循环少一部分。

第一部分是初始化部分：变量的类及其名称，该变量用于存储从列表中取出的每个对象。该对象与列表中存储的对象必须属于同一个类。

第二部分指出要遍历的列表。

下面的代码遍历structure列表，并将其中的每个对象的名字显示到屏幕上：

```
for (String name : structure) {  
    System.out.println(name);  
}
```

本章将要创建的第一个项目使用数组列表和特殊的for循环将一系列字符串按字母顺序显示到屏幕上。这些字符串来自一个数组和命令行参数。

在NetBeans中打开Java24项目，然后选择File->New File，创建一个新的Java空文件，并将其命名为StringLister。然后在源代码编辑器中输入程序清单12.1中的所有文本，并保存。

程序清单12.1 StringLister.java程序的完整源代码

```
1: package com.java24hours;
2:
3: import java.util.*;
4:
5: public class StringLister {
6:     String[] names = { "Spanky", "Alfalfa", "Buckwheat",
7:         "Daria",
8:         "Stymie", "Marianne", "Scotty", "Tommy", "Chubby" };
9:
10:    public StringLister(String[] moreNames) {
11:        ArrayList<String> list = new ArrayList<String>();
12:        for (int i = 0; i < names.length; i++) {
13:            list.add(names[i]);
14:        }
15:        for (int i = 0; i < moreNames.length; i++) {
16:            list.add(moreNames[i]);
17:        }
18:        Collections.sort(list);
19:        for (String name : list) {
20:            System.out.println(name);
21:        }
22:    }
23:
24:    public static void main(String[] arguments) {
25:        StringLister lister = new StringLister(arguments);
26:    }
27: }
```

在运行该应用程序之前，应该先选择**Run->Set Project Configuration->Customize**命令，将**main class**设置为**StringLister**，并将参数设置为名字，当有多个名字时，需要使用空格分开，比如**Jackie Mickey Farina Woim**。然后再选择**Run->Run Project**来查看结果。

命令行中指定的名字将添加到第6~7行的字符串数组中。由于在程序运行之前无法得知名字的个数，因此使用一个数组列表来存储这些字符串，这要比使用数组更合适。

使用Collection类的一个方法对存储在列表中的字符串按字母顺序排序:

```
Collections.sort(list);
```

与ArrayList一样, 这个类也位于java.util包中。在Java中, 数组列表以及其他有用的数据结构被称为集合。

当运行该程序时, 输出结果应该是一组按字母排序的名字(见图12.2)。数组列表在存储空间上的灵活性可以让你将额外的名字添加到数据结构中, 并与其他名字一起排序。

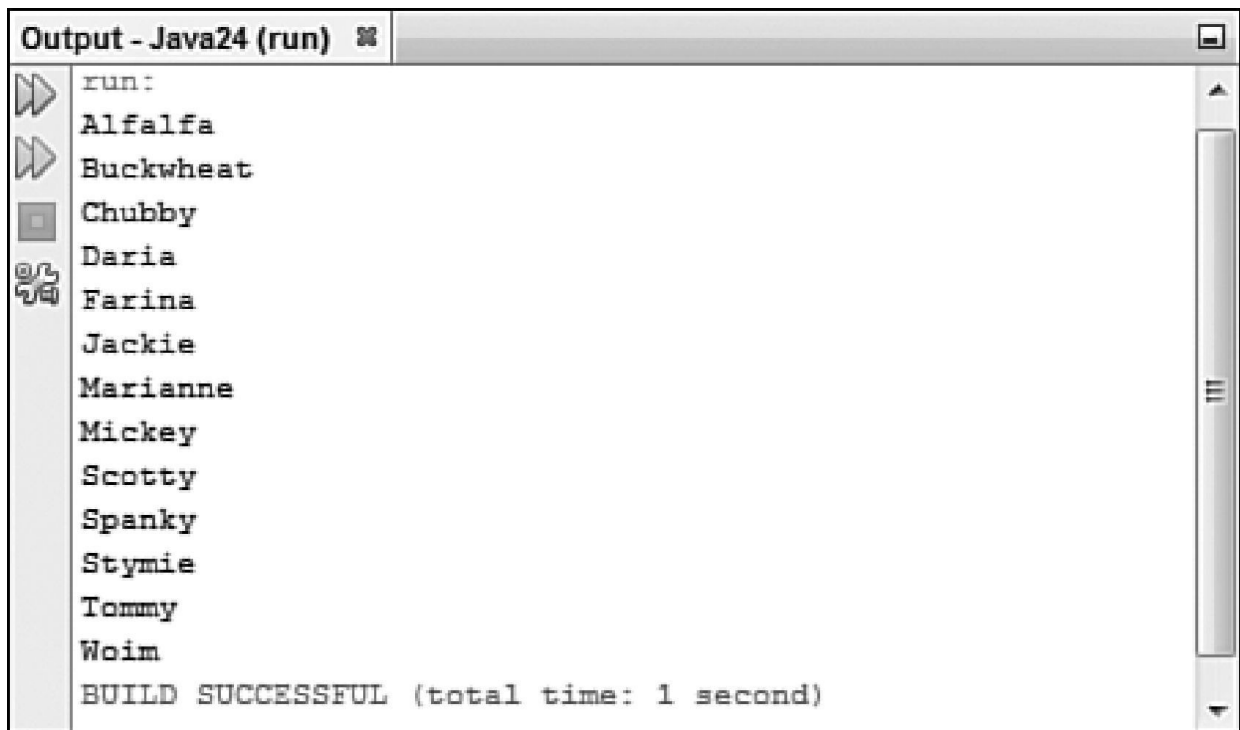


图12.2 StringLister程序的输出结果

12.5 创建子类

为了看一个关于继承的示例，现在创建一个名为**Point3D**的类，它代表三维空间中的一个点。二维点可用坐标 (x, y) 表示，而三维空间添加第三个坐标，可将其称为 z 。

Point3D对象需要做下面3件事：

- 记录对象的 (x, y, z) 坐标；
- 将对象移到新坐标 (x, y, z) 处；
- 沿 x 、 y 和 z 轴移动特定的距离。

Java有一个表示二维点的标准类，名为**Point**。

它有两个整型变量： x 和 y ，用于存储**Point**对象的 (x, y) 坐标。它还有名为**move()**的方法，用于将一个点放在指定的位置，还有一个名为**translate()**的方法，将对象沿 x 和 y 轴移动特定的距离。

在NetBeans中的Java24项目中，创建一个新的Java空文件，将其命名为**Point3D**，然后在源代码编辑器窗口中输入程序清单12.2中的全部文本，并保存。

程序清单12.2 **Point3D.java**的完整源代码

```
1: package com.java24hours;
2:
3: import java.awt.*;
4:
5: public class Point3D extends Point {
6:     public int z;
7: }
```

```
8:      public Point3D(int x, int y, int z) {
9:          super(x,y);
10:         this.z = z;
11:     }
12:
13:     public void move(int x, int y, int z) {
14:         this.z = z;
15:         super.move(x, y);
16:     }
17:
18:     public void translate(int x, int y, int z) {
19:         this.z += z;
20:         super.translate(x, y);
21:     }
22: }
```

Point3D类没有**main()**块语句，所以不能将其作为一个应用程序来运行，但是你可以在需要三维空间点的**Java**程序中使用它。

Point3D类只需完成其超类**Point**不能完成的工作，主要是记录整型变量**z**，将**z**作为**move()**方法、**translate()**方法和**Point3D()**构造函数的参数。

这些方法都使用关键字**super**和**this**。**this**用于引用当前的**Point3D**对象，因此第10行的**this.z=z;**语句将对象变量**z**设置为第8行作为参数传入的**z**的值。

super语句引用当前对象的超类**Point**，用于设置**Point3D**类继承的变量和调用继承的方法。第9行的语句**super(x, y)**调用超类的构造函数**Point(x, y)**，该构造函数用来设置**Point3D**对象的（**x, y**）坐标。由于**Point**类能够处理坐标轴**x**和**y**，如果**Point3D**类也这样做将多余。

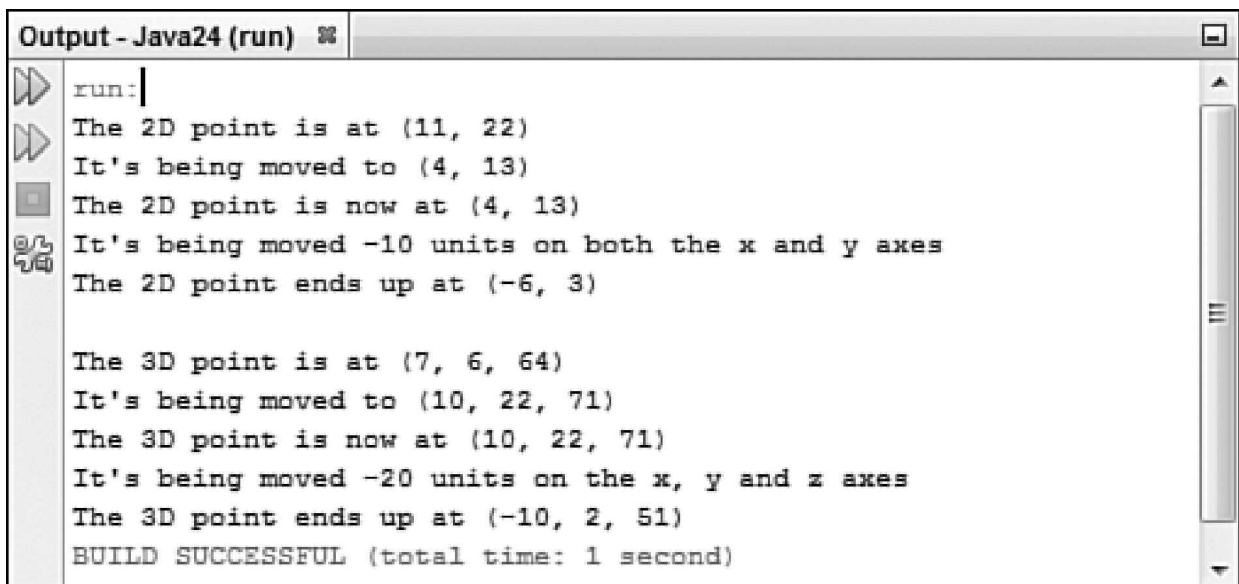
要测试新的Point3D类，可以创建一个使用Point对象和Point3D对象的程序，并在屏幕上移动它们。在NetBeans中创建一个名为PointTester的新文件，然后输入程序清单12.3中的全部文本。当保存该文件时，它将自动编译。

程序清单12.3 PointTester.java程序的完整源代码

```
1: package com.java24hours;
2:
3: import java.awt.*;
4:
5: class PointTester {
6:     public static void main(String[] arguments) {
7:         Point location1 = new Point(11,22);
8:         Point3D location2 = new Point3D(7,6,64);
9:
10:        System.out.println("The 2D point is at (" + location1.x
11:            + ", " + location1.y + ")");
12:        System.out.println("It's being moved to (4, 13)");
13:        location1.move(4,13);
14:        System.out.println("The 2D point is now at (" +
location1.x
15:            + ", " + location1.y + ")");
16:        System.out.println("It's being moved -10 units on both
the x "
17:            + "and y axes");
18:        location1.translate(-10,-10);
19:        System.out.println("The 2D point ends up at (" +
location1.x
20:            + ", " + location1.y + ")\n");
21:
22:        System.out.println("The 3D point is at (" + location2.x
23:            + ", " + location2.y + ", " + location2.z + ")");
24:        System.out.println("It's being moved to (10, 22, 71)");
25:        location2.move(10,22,71);
26:        System.out.println("The 3D point is now at (" +
location2.x
27:            + ", " + location2.y + ", " + location2.z + ")");
28:        System.out.println("It's being moved -20 units on the
x, y "
29:            + "and z axes");
30:        location2.translate(-20,-20,-20);
31:        System.out.println("The 3D point ends up at (" +
location2.x
```

```
32:         + ", " + location2.y + ", " + location2.z + "));  
33:     }  
34: }
```

如果该程序已经成功编译，通过选择**Run->Run File**运行该文件时，则会看到如图12.3所示的输出。否则，查看源代码编辑器窗口旁边的红色图标，它可以指示是哪一行代码触发了错误。



```
Output - Java24 (run) %  
run:  
The 2D point is at (11, 22)  
It's being moved to (4, 13)  
The 2D point is now at (4, 13)  
It's being moved -10 units on both the x and y axes  
The 2D point ends up at (-6, 3)  
  
The 3D point is at (7, 6, 64)  
It's being moved to (10, 22, 71)  
The 3D point is now at (10, 22, 71)  
It's being moved -20 units on the x, y and z axes  
The 3D point ends up at (-10, 2, 51)  
BUILD SUCCESSFUL (total time: 1 second)
```

图12.3 PointTester程序的输出

12.6 总结

当人们谈论生育奇迹时，他们谈论的可能不是从Java超类派生子类的方式或在类层次结构中继承行为和属性的方式。

如果现实世界与面向对象编程的工作方式相同，则莫扎特的每个子孙都可以选择成为一名卓越的作曲家，马克吐温的所有子孙都能写

出密西西比河边生活的诗歌。所有从祖先那里继承的技能都可以由你直接支配，不需要通过任何努力。

从奇迹的程度看，继承不能同物种延续和连续无安打的棒球比赛相比。然而，这是一种设计软件的高效方法，可最大限度地减少重复工作。

12.7 问与答

问：到目前为止，所创建的大多数Java程序都没有使用`extends`来从超类继承，这是否就意味着它们不在层次结构中？

答：使用Java创建的所有类都是Java类层次结构的一部分，因为编写程序时如果不使用关键字`extends`，则默认超类为Object对象。所有类的方法`equals()`和`toString()`都是自动从Object类继承而来的行为。

12.8 测验

为测试读者从本章继承了什么知识，请回答下列问题。

12.8.1 问题

1. 如果在子类中要以不同于超类的方式处理方法，应该如何做？
 - a. 删除超类中的方法。
 - b. 覆盖超类中的方法。

c. 给《San Jose Mercury News》的编辑写信，希望Java的开发者能看到这封信。

2. 哪个方法用于检索存储在数组列表中的元素？

a. `get()`。

b. `read()`。

c. `elementAt()`。

3. 可以使用哪个关键字来引用当前对象的方法和变量？

a. `this`。

b. `that`。

c. `theOther`。

12.8.2 答案

1. b. 由于可以覆盖方法，所以无需修改超类的任何方面。

2. a. 方法`get()`接受一个参数—元素的索引号。

3. a. `this`关键字引用当前对象。

12.9 练习

如果你脑海中出现了学习更多知识的强烈愿望，可以通过下面的练习来获得更多有关继承的知识。

- 创建一个**Point4D**类，在**Point3D**类创建的（**x, y, z**）坐标系的基础上再加上**t**坐标。**t**坐标代表时间，因此必须确保它不会被设置为负值。
- 使用足球队的进攻成员——边锋、外接手、阻挡线队员、跑锋、四分卫，设计一个代表这些球员技能类层次结构，将通用技能放在层次结构的较上层。例如，阻挡行为应由边锋和阻挡线队员继承，速度应由外接手和跑锋继承。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第13章 创建简单的用户界面

本章介绍如下内容：

- 创建按钮这样的用户界面组件；
- 创建标签、文本框和其他组件；
- 将组件编组；
- 将组件放在其他组件中；
- 水平和垂直滚动组件；
- 打开和关闭窗口。

在本章，所有的事情将变得繁琐起来。读者在使用Java创建第一个图形用户界面（GUI）时，将会面临很大的一个烂摊子。

计算机用户期望其软件有图形用户界面、接收来自鼠标的用户输入并像其他程序那样工作。尽管有些用户仍然在命令行环境下（比如MS-DOS或Linux shell）工作，但是大多数用户会非常不适应不提供点选、拖放图形界面的软件。

Java通过使用Swing来支持这种类型的软件，Swing是Java类的集合，它可以表示所有不同的按钮、文本框、滑块，以及可以成为GUI一部分的其他组件，同时它还包括了可以从上述组件中接收用户输入的类型。

在本章和后续章节，读者将使用Java创建并组织一个GUI。在学习过第15章以后，你就可以让这些界面接收鼠标点击和其他用户输入。

13.1 Swing和抽象窗口工具包

Java是一种跨平台语言，使用它可以为很多不同的操作系统编写程序，因此其图形用户软件必须是灵活的，不仅能够支持Windows和Mac窗口风格，还必须能够满足其他平台的要求。

在Java中，使用两组类来开发程序的用户界面：Swing和早期的称为抽象窗口工具包（Abstract Windowing Toolkit, AWT）的一组类。这些类让你能够创建图形用户界面以及接收用户输入。

Swing提供了编写使用图形用户界面的程序所需的一切。使用Java的用户界面类，可以创建包括下述元素的GUI：

- 按钮、复选框、标签和其他简单组件；
- 文本框、滑块和其他复杂组件；
- 下拉菜单和弹出菜单；
- 窗口、框架、对话框、面板和applet窗口。

By the Way

注意

Swing包含了大量可以自定义的组件，而且其自定义的方式远比这里讲解的多。本书涵盖了大多数常见的组件以及相应的最有用的方法。在接下来的4章中，你将学习到与每个组件相关的更多内容，并在Oracle发布的

Java类库官方文档（地址为<http://download.java.net/jdk8/docs/api>）中找到新的方法。它还针对Java 8中的每一个类和接口提供了额外的参考文档。

13.2 使用组件

在Java中，图形用户界面的每部分都由Swing包中的一个类表示。对于按钮，是JButton类，窗口是JWindow类，文本框是JTextField类等。

为创建并显示界面，需要创建对象，设置其变量并调用其方法。使用的技巧与前面介绍面向对象编程的三章中相同。

组织图形用户界面时，需要使用两类对象：组件和容器。组件是用户界面中的独立元素，如按钮或滑块；容器是用于容纳其他组件的组件。

创建界面的第一步是创建能够容纳组件的容器。在应用程序中，该容器通常是窗口或框架。

13.2.1 窗口和框架

窗口和框架是能够在用户界面上显示，并且可以容纳其他组件的容器。窗口是一种简单的容器，不像常规图形用户界面那样，在顶端有标题栏和其他按钮。框架是一种这样的窗口：包含用户运行软件时希望看到的所有常见的窗口特性，如关闭按钮、最大化按钮和最小化按钮。

这些容器分别是使用Swing包中的JWindow和JFrame类创建的。为了在Java程序中引用Swing包而且无需使用完整的包和类名，可使用下面的语句：

```
import javax.swing.*;
```

这条语句只是从javax.swing包中导入了类的名字。在Swing包中还有其他可用的包。

一种在Java应用程序中使用框架的方法是，将应用程序声明为JFrame的子类，这样应用程序将继承作为框架所需的行为。下面的语句创建了JFrame的一个子类：

```
import javax.swing.*;

public class MainFrame extends JFrame {
    public MainFrame() {
        // set up the frame
    }
}
```

这个类创建了一个框架，但没有完整地设置。创建框架时，必须在框架的构造函数中执行几种操作：

- 调用超类JFrame的构造函数；
- 设置框架的标题；

- 设置框架的大小;
- 设置框架的外观;
- 定义用户关闭框架时应执行的操作。

还必须使框架可见，除非由于某种原因，在应用程序开始运行时不应显示框架。

所有这些操作都可以在框架的构造函数中完成。该方法首先应使用关键字`super`调用`JFrame`的一个构造函数。下面是一个例子：

```
super();
```

该语句调用无参数的`JFrame`构造函数。也可以调用将框架标题作为参数的构造函数：

```
super("Main Frame");
```

这将框架的标题设置为指定的字符串，标题出现框架顶端的标题栏中。在这里，标题为**Main Frame**。

如果不采用这种方式设置标题，可以调用框架的`setTitle()`方法，并将一个字符串作为参数：

```
setTitle("Main Frame");
```

要指定框架的大小，可调用`setSize(int, int)`方法并指定两个参数：宽度和高度。下面的语句将框架设置为宽350像素，高125像素：

```
setSize(350, 125);
```

设置框架大小的另一种方式是，先用组件填充它，然后调用不带参数的`pack()`方法，如下例所示：

```
pack();
```

方法`pack()`根据框架中每个组件的首选尺寸来设置框架。每个界面组件都有首选尺寸，虽然有时会忽略首选尺寸，这取决于组件在界面中是如何排列的。调用`pack()`方法前，无需显式地设置框架的大小，该方法在框架显示前将其设置为足够大。

每个框架的标题栏上都有一个用于关闭框架的按钮。在Windows系统中，该按钮出现在框架的右上角并用“X”标识。要定义单击该按钮时发生的情况，可调用框架的`setDefaultCloseOperation(int)`方法，并将4个`JFrame`类变量之一作为参数。

- **EXIT_ON_CLOSE**: 按钮被单击时退出程序。
- **DISPOSE_ON_CLOSE**: 关闭框架，同时继续运行应用程序。
- **DO_NOTHING_ON_CLOSE**: 保持框架为打开状态并继续运行。
- **HIDE_ON_CLOSE**: 关闭框架并继续运行。

下面这种方法调用时最常见的，因为当关闭一个应用程序的图形用户界面时，这意味着应用程序应该完成其工作并关闭。

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

当使用**Swing**创建图形用户界面时，可以对它的外观（一种视觉主题）进行自定义，从而控制按钮和其他组件的显示方式和行为。

Java 包含了一个增强的外观，名为**Nimbus**，当在类中使用时，必须先启用。通过调用主**Swing**包中**UIManager**类的**setLookAndFeel()**方法可以设置外观。

该方法接受一个参数：外观类的完整名称。

下面的语句将**Nimbus**设置为外观：

```
UIManager.setLookAndFeel(  
    "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"  
);
```

最后要做的一件事情是显示框架：使用`true`作为参数调用`setVisible()`方法：

```
setVisible(true);
```

这将以定义的宽度和高度打开框架。也可以用`false`作为参数调用该方法，以停止显示该框架。

程序清单13.1包含了本节介绍的源代码，在Java空文件中输入这些语句并将文件保存为`SalutonFrame.java`。

程序清单13.1 SalutonFrame.java程序

```
1: package com.java24hours;
2:
3: import javax.swing.*;
4:
5: public class SalutonFrame extends JFrame {
6:     public SalutonFrame() {
7:         super("Saluton mondo!");
8:         setLookAndFeel();
9:         setSize(350, 100);
10:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11:        setVisible(true);
12:    }
13:
14:    private void setLookAndFeel() {
15:        try {
16:            UIManager.setLookAndFeel(
17:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
18:            );
19:        } catch (Exception exc) {
20:            // ignore error
21:        }
22:    }
```

```
23:
24:     public static void main(String[] arguments) {
25:         SalutonFrame frame = new SalutonFrame();
26:     }
27: }
```

在程序清单13.1中，第24~26行包含了一个main()方法，它使该框架类变为一个应用程序。当运行该程序时，将会看到图13.1所示的框架。

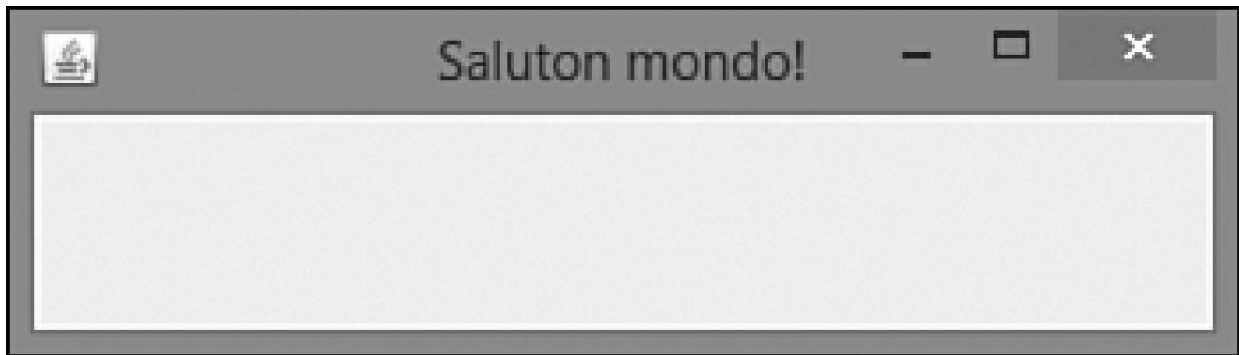


图13.1 在应用程序中显示框架

程序SalutonFrame只是显示一个标题：世界语问候“Saluton mondo!”。该框架是一个空窗口，因为它还没有包含任何组件。

要在框架中添加组件，必须创建组件并将其加入到容器中。每个容器都有一个add()方法，该方法接受一个参数：要显示的组件。

SalutonFrame类包含一个setLookAndFeel()方法，它可以将Nimbus指定为框架的外观。代码第16~18行调用了UIManager类的这个setLookAndFeel()方法来实现该目的。

调用该方法的语句放置在try-catch语句块中，从而可以处理有可能发生的错误。try和catch语句在前面一直没有介绍，因此对读者来说还较为陌生。使用这些语句来处理错误的知识将在第18章详细讲解。

此时，你只需要知道调用UIManager.setLookAndFeel()方法可以设置GUI的外观即可。如果调用该方法时发生了错误，程序将显示其默认的外观，而不是Nimbus。

Did you Know?

提示

在程序清单13.1中可能有些东西之前没有见过。仔细看第16~18行，这其实是一条语句，但是跨了三行。该语句可以放在一行中，这里之所以这样处理，是为了让程序更具可读性。Java编译器并不会关注多出来的空格，只要语句没有问题，而且以分号结尾，就可以放在多行显示。

13.2.2 按钮

可以添加到容器中的一种简单组件是JButton对象。JButton和本章介绍的其他组件一样，也位于java.awt.swing包中。JButton对象是一个可单击的按钮，其标签描述了单击按钮的结果。该标签可以是文本、图像或两者的组合。下面的语句创建一个名为okButton的JButton对象，并将其文本标签设置为OK：

```
JButton okButton = new JButton("OK");
```

创建JButton等组件后，应调用add()方法将其加入到容器中：

```
add(okButton);
```

在容器中添加组件时，不需要指明组件在容器中显示的位置，组件的布局由被称为布局管理器的对象决定。最简单的布局管理器是FlowLayout类，它位于java.awt包中。

要让容器使用特定的布局管理器，必须首先创建该布局管理器的对象。通过调用无参数的构造函数可以创建FlowLayout对象，语句如下所示：

```
FlowLayout flo = new FlowLayout();
```

创建布局管理器后，调用容器的setLayout ()方法将其管理器同容器关联起来，如下所示：

```
setLayout(flo);
```

这条语句将flo对象指定为容器布局管理器。

接下来要创建的Java应用程序是一个名为Playback的类，它可以显示一个包含3个按钮的框架。在一个Java空文件中输入程序清单13.2中的所有文本，然后保存。

程序清单13.2 Playback.java的完整源代码

```
1: package com.java24hours;
2:
3: import javax.swing.*;
4: import java.awt.*;
5:
6: public class Playback extends JFrame {
7:     public Playback() {
8:         super("Playback");
9:         setLookAndFeel();
10:        setSize(225, 80);
11:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12:        FlowLayout flo = new FlowLayout();
13:        setLayout(flo);
14:        JButton play = new JButton("Play");
15:        JButton stop = new JButton("Stop");
16:        JButton pause = new JButton("Pause");
17:        add(play);
18:        add(stop);
19:        add(pause);
20:        setVisible(true);
21:    }
22:
23:    private void setLookAndFeel() {
24:        try {
25:            UIManager.setLookAndFeel(
26:                "
com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
27:            );
28:        } catch (Exception exc) {
29:            // ignore error
30:        }
31:    }
32:
33:    public static void main(String[] arguments) {
34:        Playback frame = new Playback();
35:    }
36: }
```

Playback程序在代码第12行创建了一个FlowLayout布局管理器，并在代码第13行将其用于框架。当在代码第17~19行将3个按钮添加到框架中时，它们将由该布局管理器进行放置。

当运行该程序时，其输出结果应该如图13.2所示。尽管用户可以单击其中的每个按钮，但什么也不会发生，因为该程序没有包含任何接收并响应用户输入的方法，这种方法将在第15章介绍。

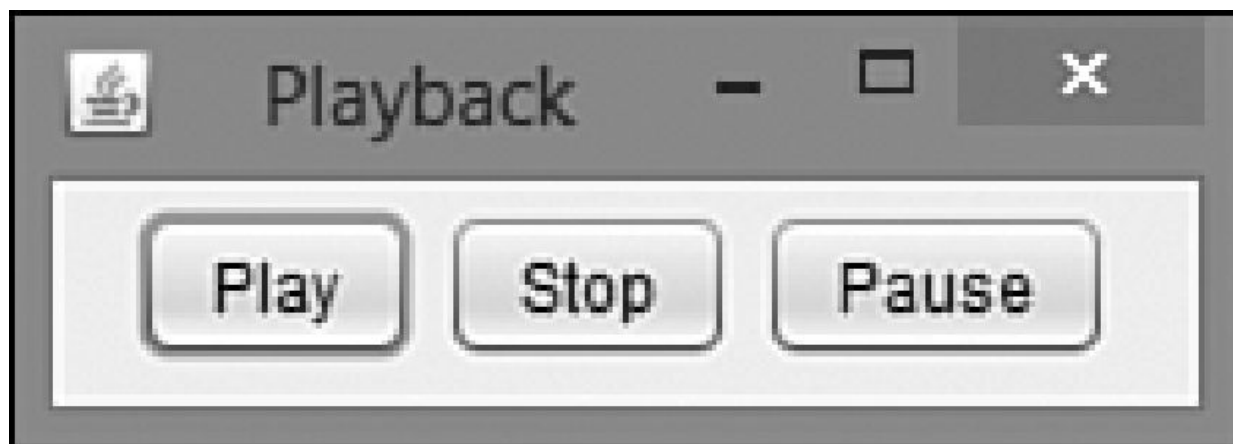


图13.2 在GUI上显示按钮

可以通过这种方法将多个Swing用户组件添加到容器中。

By the Way

注意

本章将介绍很多不同种类的用户界面组件，这里将不会列出创建每个组件的源代码。读者可在本书的配套网站www.java24hours.com找到本章所有程序的源代码。

13.2.3 标签和文本框

JLabel组件显示用户不能修改的信息，这种信息可以是文本、图形或两者的组合。这些组件常用于标识界面中的其他组件，因此而得名。它们常用于标识文本框。

TextField组件是用户可以输入单行文本的区域。创建文本框时，可以设置其宽度。

下面的语句创建了一个**JLabel**组件和一个**TextField**对象，并将它们加入到容器中：

```
JLabel pageLabel = new JLabel("Web page address: ", JLabel.RIGHT);
TextField pageAddress = new TextField(20);
FlowLayout flo = new FlowLayout();
setLayout(flo);
add(pageLabel);
add(pageAddress);
```

图13.3并排地显示该标签和文本框。这里的两条语句都使用一个参数来设置组件的外观。



图13.3 显示标签和文本框

标签pageLabel的文本被设置为“Web page address:”，并使用了参数JLabel.RIGHT。该参数将文本与标签右对齐，JLabel.LEFT将文本左对齐，而JLabel.CENTER居中显示文本。用于JTextField的参数指定文本框的宽度应该大约为20个字符，也可以使用下面这样的语句指定显示在文本框的默认文本：

```
JTextField country = new JTextField("Togolese Republic", 29);
```

该语句创建了一个JTextField对象，其宽度为20个字符，且内容默认为“US”。

对象包含的文本可使用方法getText()来检索，它返回一个字符串：

```
String countryChoice = country.getText();
```

读者可能猜到了，也可以使用相应的方法来设置文本：

```
country.setText("The People's Republic of China");
```

该语句将文本设置为“The People’s Republic of China”。

13.2.4 复选框

JCheckBox组件由一行文本和方框组成，用户可以选中它，也可以不选中。下面的语句创建一个**JCheckBox**对象并将其加入到一个容器中：

```
JCheckBox jumboSize = new JCheckBox("Jumbo Size");
FlowLayout flo = new FlowLayout();
setLayout(flo);
add(jumboSize);
```

构造函数**JCheckBox()**的参数指定了显示在复选框旁边的文本。如果要选中该复选框，可使用下面的语句：

```
JCheckBox jumboSize = new JCheckBox("Jumbo Size", true);
```

JCheckBox可以单个显示，也可以编成组。在一组复选框中，一次只能选中一个。要使**JCheckBox**对象成为某个组的一部分，必须创建一个**ButtonGroup**对象，请看下面的语句：

```
JCheckBox frogLegs = new JCheckBox("Frog Leg Grande", true);
JCheckBox fishTacos = new JCheckBox("Fish Taco Platter", false);
JCheckBox emuNuggets = new JCheckBox("Emu Nuggets", false);
FlowLayout flo = new FlowLayout();
ButtonGroup meals = new ButtonGroup();
meals.add(frogLegs);
meals.add(fishTacos);
meals.add(emuNuggets);
setLayout(flo);
add(jumboSize);
add(frogLegs);
```

```
add(fishTacos);  
add(emuNuggets);
```

上述语句创建了3个复选框，并将它们都编组到名为meals的ButtonGroup对象下。最初，复选框Frog Leg Grande被选中，但如果用户选择其他meals复选框，复选框Frog Leg Grande的选中标记将消失。图13.4显示了本节介绍的复选框。

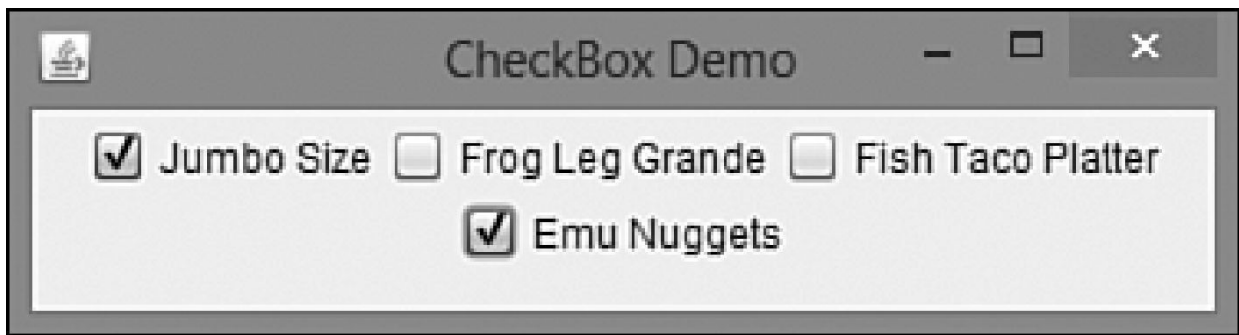


图13.4 显示复选框组件

13.2.5 组合框

JComboBox组件是一个弹出式选择列表，也可以设置成能够接收文本输入。在这种情况下，用户可以使用鼠标选中列表项，也可以使用键盘来输入文本。组合框有点像一组复选框，但在没有打开列表时，只能看见其中的一个选项。

要创建一个JComboBox对象，必须在创建这种对象后加入每个选项，如下所示：

```
JComboBox profession = new JComboBox();
```



```
FlowLayout flo = new FlowLayout();  
profession.addItem("Butcher");  
profession.addItem("Baker");  
profession.addItem("Candlestick maker");  
profession.addItem("Fletcher");  
profession.addItem("Fighter");  
profession.addItem("Technical writer");  
setLayout(flo);  
add(profession);
```

该示例创建了一个JComboBox组件，其中包含6个可供用户选择的选项。被选中的选项出现在该组件的可视区域。图13.5显示了弹出式列表被打开时的情况。

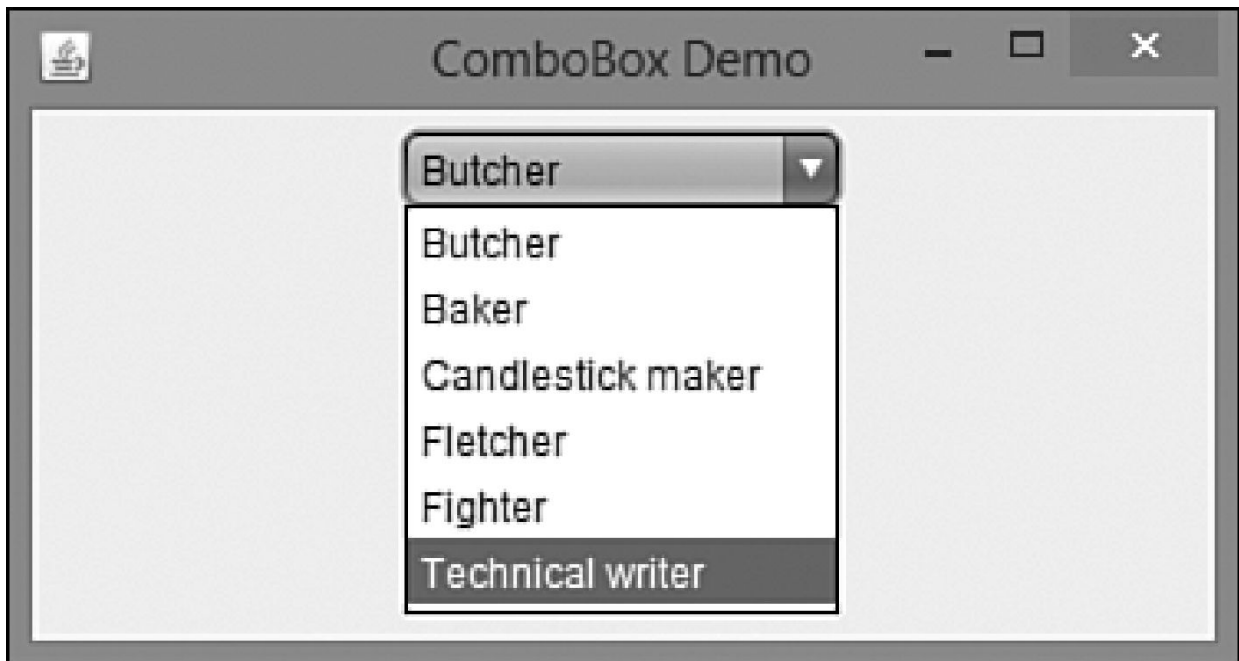


图13.5 显示组合框组件

要让JComboBox组件能够接收文本输入，必须使用true作为参数调用其setEditable()方法，如下面的语句所示：

```
profession.setEditable(true);
```

必须在将组件加入到容器中之前调用该方法。

13.2.6 文本区域

`JTextArea`组件允许用户输入多行文本，你可以指定该组件的宽度和高度。下面的语句创建了一个`JTextArea`组件，其宽度大约为40个字符，高度为8行，然后将该组件加入到容器中：

```
JTextArea comments = new JTextArea(8, 40);  
FlowLayout flo = new FlowLayout();  
setLayout(flo);  
add(comments);
```

图13.6显示了将该组件加入到框架中的情况。

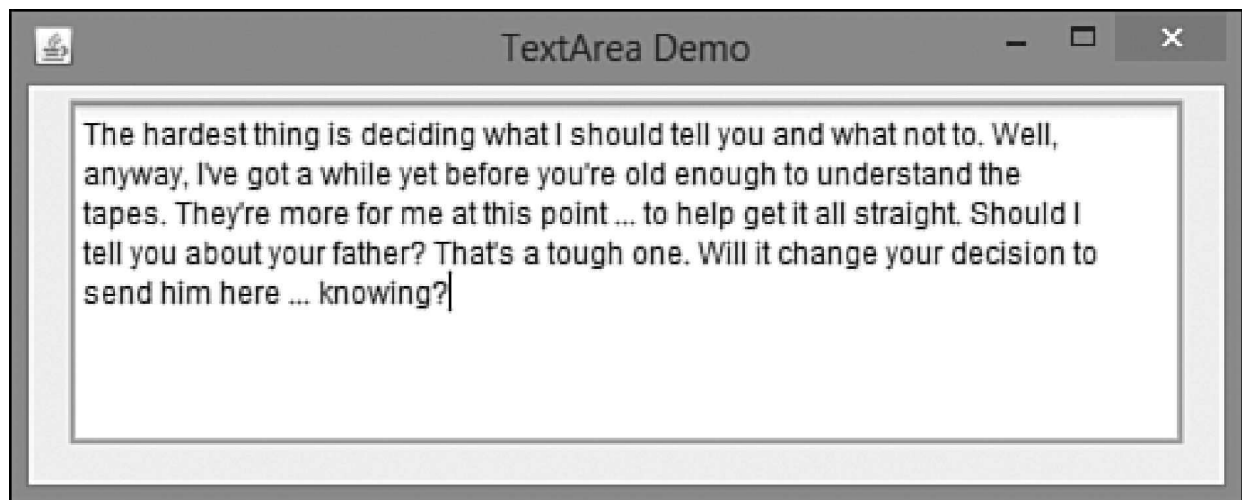


图13.6 显示文本区域组件

可以在构造函数`JTextArea()`中指定要在文本区域显示的字符串，可以使用换行符“`\n`”进行换行，如下所示：

```
JTextArea comments = new JTextArea("I should have been a pair\n"
    + "of ragged claws.", 10, 25);
```

当用户输入的文本超出一行时，可以调用文本区域的两个方法来指定组件的处理方式。调用带有`true`参数的`setLineWrap (boolean)`方法可以让文本换行：

```
comments.setLineWrap(true);
```

要确定文本如何换到下一行，可以使用`setWrapStyleWord (boolean)`方法。当使用`true`参数调用它时，文本将基于单词结束的位置来换行；使用`false`参数调用时，文本将基于字符来换行。

如果不调用这些方法，则文本不会换行。用户可以在同一行不断输入文本，从而超出文本区域的右边缘，直到按下`Enter`键。

文本区域组件以你可能想不到的方式来运行：当用户输入的文本到达区域的底部时，它会自动增大区域，但是在区域的右边缘或下边

缘并没有滚动条。要想用更好的方式来实现文本区域组件，必须将其放置到名为滚动面板的容器中。

GUI中的组件通常要比显示它的区域大。为了让显示区域能够从组件的一部分移动到另外一部分，可以使用垂直滚动条和水平滚动条。

在Swing中，可以将组件添加到一个滚动面板中来实现滚动。滚动面板是由JScrollPane类表示的一个容器。

可以使用下述构造函数来创建滚动面板。

- `JScrollPane ()`——创建一个在需要时才出现水平滚动条和垂直滚动条的滚动面板。
- `JScrollPane (int, int)`——创建一个带有指定垂直滚动条和水平滚动条的滚动面板。
- `JScrollPane (Component)`——创建一个包含指定用户界面组件的滚动面板。
- `JScrollPane (Component, int, int)`——创建一个带有指定组件、指定垂直滚动条和水平滚动条的滚动面板。

这些构造函数中的整型参数决定了如何在面板中使用滚动条。可以使用下面的类变量作为参数：

- `JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED`或
`JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED`
- `JScrollPane.VERTICAL_SCROLLBAR_NEVER`或
`JScrollPane.HORIZONTAL_SCROLLBAR_NEVER`

- JScrollPane.VERTICAL_SCROLLBAR_ALWAYS或JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS

如果创建了一个不带组件的滚动面板，可以使用面板的add(Component)方法来添加组件。在设置完一个滚动面板后，它应该添加了一个容器（而不是一个组件）。

下面是对前面示例的重写，它将文本区域放到了滚动面板中。

```
FlowLayout flo = new FlowLayout();
setLayout(flo);
JTextArea comments = new JTextArea(8, 40);
comments.setLineWrap(true);
comments.setWrapStyleWord(true);
JScrollPane scroll = new JScrollPane(comments,
    JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
    JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
add(scroll);
```

图13.7为该示例的显示结果。

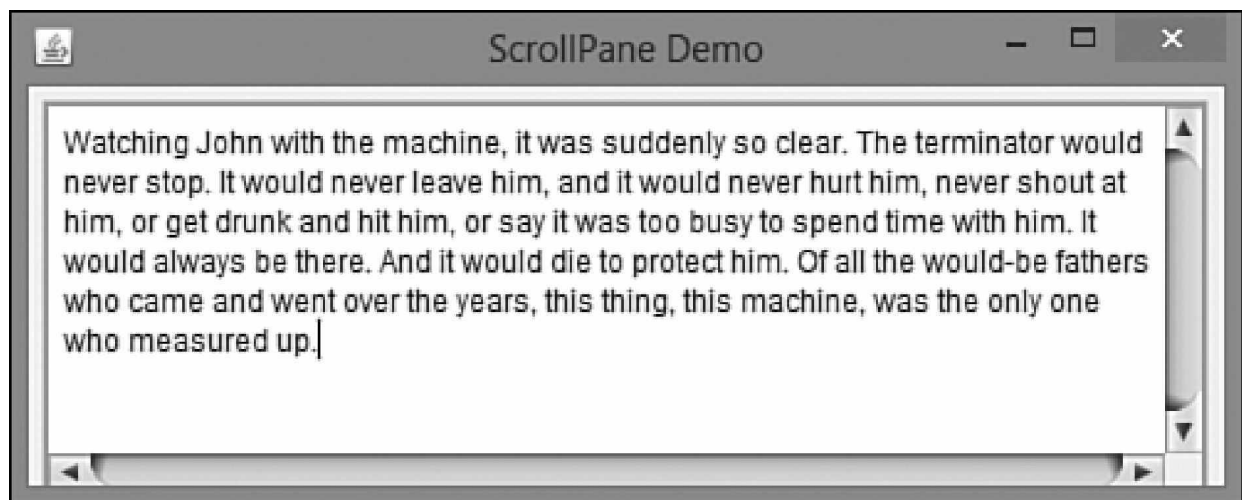


图13.7 在一个滚动面板容器中显示文本区域组件

13.2.7 面板

本章要创建的最后一种组件是面板，它是使用Swing中的JPanel类创建的。JPanel对象是可在图形用户界面中使用的最简单的一种容器，用于将显示区域划分成不同的组件组。将显示区域分成几部分后，可以在每个部分使用不同的布局管理器。

可使用下面的语句来创建JPanel对象并将其指派到布局管理器中：

```
JPanel topRow = new JPanel();  
FlowLayout flo = new FlowLayout();  
topRow.setLayout(flo);  
add(topRow);
```

当在一个界面中排列组件时，经常会用到面板，这将在第14章介绍。

通过调用面板的add()方法可以将组件添加到面板中。还可以通过调用setLayout()方法，直接给面板指派一个布局管理器。

当需要在界面中包含绘画区域时（如显示图像文件中的图像），也可以使用面板。

JPanel的另一种用途是，用于创建可加入到其他类中的组件，这将在本章后面介绍。

13.3 创建自己的组件

面向对象编程的一个主要优点是，能够在不同的项目中重用类。在接下来的程序中，读者将创建一个特殊的面板组件，可在其他Java程序中重用它。该组件名为FreeSpacePanel，它用于报告运行该应用程序的计算机的可用磁盘空间。这个程序可以显示可用的磁盘空间、总的磁盘空间，以及两者的百分比。

创建用户界面组件的第一步是，确定继承哪个现有组件。你的新组件将包括现有组件的所有属性和行为，以及你修改和添加的相应内容。

程序清单13.3中定义的FreeSpacePanel组件是JPanel的一个子类。在一个新的Java空文件中输入程序清单13.3中的全部文本，并保存。

程序清单13.3 FreeSpacePanel.java完整的源代码

```
1: package com.java24hours;
2:
3: import java.io.IOException;
4: import java.nio.file.*;
5: import javax.swing.*;
6:
7: public class FreeSpacePanel extends JPanel {
8:     JLabel spaceLabel = new JLabel("Disk space: ");
9:     JLabel space = new JLabel();
10:
11:     public FreeSpacePanel() {
12:         super();
13:         add(spaceLabel);
14:         add(space);
15:         try {
16:             setValue();
17:         } catch (IOException ioe) {
18:             space.setText("Error");
```

```

19:         }
20:     }
21:
22:     private final void setValue() throws IOException {
23:         // get the current file storage pool
24:         Path current = Paths.get("");
25:         FileStore store = Files.getFileStore(current);
26:         // find the free storage space
27:         long totalSpace = store.getTotalSpace();
28:         long freeSpace = store.getUsableSpace();
29:         // get this as a percentage (with two digits)
30:         double percent = (double)freeSpace /
(double)totalSpace * 100;
31:         percent = (int)(percent * 100) / (double)100;
32:         // set the label's text
33:         space.setText(freeSpace + " free out of " + totalSpace
+ " ("
34:                     + percent + "%)");
35:
36:     }
37: }

```

当想要显示与可用磁盘空间相关的信息时，可以将这个类添加到任何图形用户界面中。这个类无法作为一个应用程序自行运行。

`FreeSpacePanel`中的`setValue()`方法将标签的文本设置为表示磁盘空间的信息。在代码第22行中声明该方法时，还带有**final**关键字：

```

private final void setValue() {
    // ...
}

```

该方法使用了**final**关键字后可以防止它被子类中的方法覆盖。作为GUI组件，这对`FreeSpacePanel`来说是必需的。

第11~20行的构造函数创建了面板，具体情况如下。

- 第12行： `super ()`方法调用 `JPanel`类的构造函数，确保其设置是正确的。
- 第13行： 在第8行以实例变量的形式创建了一个名为 `spaceLabel` 的新标签，并将其文本设置为“**Disk space:**”。然后将标签作为一个参数调用 `add (Component)`方法，将该标签添加到面板中。
- 第14行： 第19行创建的实例变量 `space` 标签（无文本）也添加到面板中。
- 第16行： 调用 `setValue ()`方法，设置 `space` 标签的文本。

与程序清单13.中设置外观时使用的 `try-catch` 语句块一样，第16行调用的 `setValue ()`方法放置到了另外一个 `try-catch` 语句块中。第18章将进一步讲解如何处理错误，我们这里简单一提：`try-catch`用来处理Java程序中可能发生的一个或多个错误。

这样做很有必要，因为当访问计算机的文件系统，以查看何用的磁盘空间时，调用 `setValue ()`方法可能会引发错误。错误在Java中也有专门的类——表示输入/输出错误的 `IOException`。

`try`语句块将可能引发错误的代码封装起来。`catch`语句块用来识别括号内的错误类型，并在该错误发生时执行相应代码。

如果在引用程序时用到了这样一个错误，第18行将 `space` 标签的文本设置为“**Error**”，提醒用户发生了错误。

要想让自定义组件做一些有趣的事情，则需要用到Java文件处理机制。这将在第21章中讲到。

为了计算出计算机上的可用磁盘空间，需要java.nio.file包中的4个类，用来访问计算机的文件系统。

Path对象表示文件或文件夹的位置。**Path**类有一个**get(String)**方法可以将一个字符串转换为相应的路径。当使用“”（一个空字符串）调用该方法时，返回正在运行的Java应用程序所在的当前文件夹的路径。

```
Path current = Paths("");
```

磁盘的存储信息由Java中的**FileStore**对象来表示，这是一个文件存储池。在拥有了路径之后，可以使用**Files**类中的一个方法来获得存储在该路径中的文件。调用**Files**的**getFileStore(Path)**方法，获得文件存储池。

```
FileStore store = Files.getFileStore(current);
```

有了文件存储池之后，可以调用**getTotalSpace ()**和**getUsableSpace ()**方法确定当前磁盘还有多少可用空间。

```
long totalSpace = store.getTotalSpace();  
long freeSpace = store.getUsableSpace();  
double percent = (double)freeSpace / (double)totalSpace * 100;
```

在最后一条语句中计算出来的百分比可能具有相当多的小数位，比如64.8675309。

下面这条语句将一个double型的百分比转换为不超过2位小数的数值。

```
percent = (int)(percent * 100) / (double)100;
```

该语句的前半部分将百分比乘以100（即将小数点向右移2位），然后将其转换为一个整数，比如64.8675309成为6487。

语句的后半部分将该值除以100（即将小数点向左移2位），恢复为一个百分比。这使得6486成为64.86。

在将一个百分比的小数位限制为两位后，可以设置标签的文本，以报告可用和已使用的磁盘空间：

```
space.setText(freeSpace + " free out of " + totalSpace + " (" + percent + "%");
```

在将创建的面板添加到用户图形界面中后，才能看到该面板。要测试FreeSpacePanel，可以创建一个程序清单13.4中定义的FreeSpaceFrame应用程序。输入程序清单13.4的所有文本，将其命名为FreeSpaceFrame类。

程序清单13.4 FreeSpaceFrame.java的完整源代码

```
1: package com.java24hours;
2:
3: import java.awt.*;
4: import javax.swing.*;
5:
6: public class FreeSpaceFrame extends JFrame {
7:     public FreeSpaceFrame() {
8:         super("Disk Free Space");
9:         setLookAndFeel();
10:        setSize(500, 100);
11:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12:        FlowLayout flo = new FlowLayout();
13:        setLayout(flo);
14:        FreeSpacePanel freePanel = new FreeSpacePanel();
15:        add(freePanel);
16:        setVisible(true);
17:    }
18:
19:    private void setLookAndFeel() {
20:        try {
21:            UIManager.setLookAndFeel(
22:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
23:            );
24:        } catch (Exception exc) {
25:            // ignore error
26:        }
27:    }
28:
29:    public static void main(String[] arguments) {
30:        FreeSpaceFrame frame = new FreeSpaceFrame();
31:    }
32: }
```

该应用程序在第14行创建了一个FreeSpacePanel组件，然后在第15行将面板添加到框架中。

当运行该程序时，其结果如图13.8所示（其中显示的数值与你的计算机相关）。

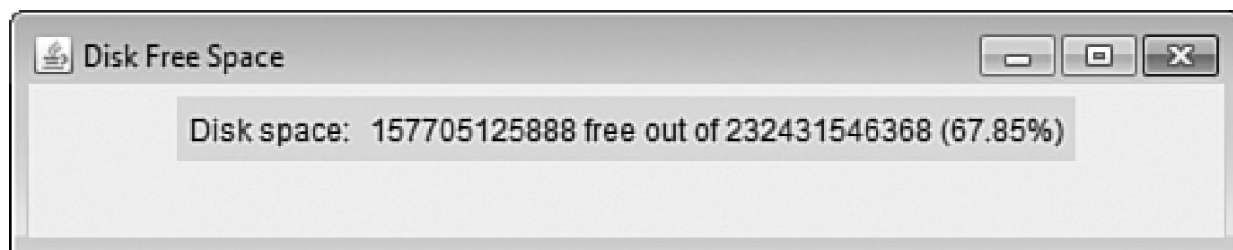


图13.8 显示一个自定义的磁盘空间组件

13.4 总结

用户期望其运行的程序有可点选的可视化界面，这一希望给创建软件提出了更严峻的挑战，Java通过Swing向程序员提供了这些功能。Swing提供了创建可运行的GUI所需的所有类，不管用户在何种操作系统中运行Java程序。

使用Swing来开发Java图形应用程序可以帮助读者练习面向对象编程。每一个组件、容器和布局管理器都使用其对象来表示，它们从同一个超类中继承了常见的行为。例如，本章介绍的所有用户界面组件都是java.swing.JComponent的子类。

下一章将介绍如何使用布局管理器来设计图形用户界面。相较于FlowLayout，布局管理器能够用更为复杂的方法指定如何在容器中放置组件。

13.5 问与答

问：如果没有给容器指定布局管理器，组件将如何排列？

答：在简单的容器（如面板）中，默认使用FlowLayout来排列组件，每一个组件像在页面中显示单词那样被加入到容器：从左到右排列，当前行没有空间后进行下一行，再按从左到右的顺序排列。下一章将讲到，框架、窗口和applet默认使用BorderLayout布局。

问：为什么很多图形用户界面的类的名字都以J打头，比如JFrame和JLabel？

答：这些类都属于Swing框架，该框架是Java类库中第二种可包含图形用户界面类的方式。抽象窗口工具包（AWT）是第一种方式，它的类名要更为简单，比如Frame和Label。

AWT类属于java.awt包及其相关包，而Swing类属于javax.swing包和相关包，因此它们有相同的类名。之所以在名字面前使用字母J是为了对两者进行区分。

Swing类也称之为Java基础类（JFC）。

13.6 测验

经过本章的艰苦学习，如果读者的头脑还没有变成GUI碎片，请回答下列问题以测试自己的技能。

13.6.1 问题

1. 下列哪种用户组件可用作容纳其他组件的容器？

- a. TupperWare °
 - b. JPanel °
 - c. Choice °
2. 在容器中必须首先做下面哪项工作？
- a. 指定布局管理器 °
 - b. 添加组件 °
 - c. 无关紧要 °
3. 哪个方法决定如何在容器中排列组件？
- a. setLayout() °
 - b. setLayoutManager() °
 - c. setVisible() °

13.6.2 答案

1. b. 可以在面板中添加组件，然后将面板加入到其他容器中，如框架 °
2. a. 必须在添加组件前指定布局管理器，这样才能正确地加入组件 °

3. a. 方法`setLayout()`接受一个参数，即决定如何排列组件的布局管理器对象。

13.7 练习

为更深入地理解GUI设计，请完成下面的练习：

- 修改应用程序`SalutonFrame`，使其在框架的主要而不是标题栏中显示文本“`Saluton Mondo!`”；
- 增强`FreeSpacePanel`组件的功能，使其在显示磁盘容量时每三位显示一个逗号。为此，可以使用`StringBuilder`对象、`String`类方法`charAt(int)`和用于迭代字符串的一个`for`循环。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第14章 用户界面的布局

本章介绍如下内容：

- 创建布局管理器；
- 为容器指定布局管理器；
- 使用面板来组织界面中的组件；
- 使用不常见的布局；
- 为Java应用程序创建原型。

开始为Java程序设计图形用户界面时，面临的一个障碍是你的组件会移动。容器大小发生变化时（如用户调整框架的大小时），容器中的组件将根据容器的新尺寸重新排列。

这种变化对程序员有利，因为它考虑到了界面组件在不同操作系统中的显示方式。对于同一个Java程序，可单击的按钮在Windows、Linux和Mac操作系统中的外观可能不同。

使用一组称为布局管理器的类来排列界面中的组件，这些类定义了组件如何在容器中显示。界面中的每个容器都有自己的布局管理器。

14.1 使用布局管理器

在Java中，组件在容器中的位置取决于组件本身的大小、其他组件的大小以及容器的宽度和高度。下列因素将影响按钮、文本框和其他组件的布局：

- 容器的大小；
- 其他组件和容器的大小；
- 使用的布局管理器。

可使用几种布局管理器来影响组件的显示。面板的默认布局管理器是java.awt包中的FlowLayout类，前一章使用的就是它。

使用FlowLayout时，像在页面中排列英文单词那样排列组件：从左到右排列，当前行没有空间后进入下一行。

当在框架中添加近组件时，它可以使用如下代码示例来调用流式布局：

```
FlowLayout topLayout = new FlowLayout();  
setLayout(topLayout);
```

也可以指定用于特定容器如JPanel对象) 的布局管理器，为此可以使用该容器对象的setLayout()方法。

创建一个新的Java空文件，将其命名为Crisis，然后输入程序清单14.1中的全部文本，并保存。该应用程序的图形用户界面包含5个按钮。

程序清单14.1 Crisis.java程序的完整源代码

```
1: package com.java24hours;
2:
3: import java.awt.*;
4: import javax.swing.*;
5:
6: public class Crisis extends JFrame {
7:     JButton panicButton;
8:     JButton dontPanicButton;
9:     JButton blameButton;
10:    JButton mediaButton;
11:    JButton saveButton;
12:
13:    public Crisis() {
14:        super("Crisis");
15:        setLookAndFeel();
16:        setSize(348, 128);
17:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18:        FlowLayout flo = new FlowLayout();
19:        setLayout(flo);
20:        panicButton = new JButton("Panic");
21:        dontPanicButton = new JButton("Don't Panic");
22:        blameButton = new JButton("Blame Others");
23:        mediaButton = new JButton("Notify the Media");
24:        saveButton = new JButton("Save Yourself");
25:        add(panicButton);
26:        add(dontPanicButton);
27:        add(blameButton);
28:        add(mediaButton);
29:        add(saveButton);
30:        setVisible(true);
31:    }
32:
33:    private void setLookAndFeel() {
34:        try {
35:            UIManager.setLookAndFeel(
36:
37:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
38:            );
39:        } catch (Exception exc) {
40:            // ignore error
41:        }
42:    }
43:
44:    public static void main(String[] arguments) {
45:        Crisis frame = new Crisis();
46:    }
```

```
46: }
```

图14.1所示为该应用程序的运行结果。



图14.1 使用流式布局来排列组件

FlowLayout类仅根据容器的尺寸来排列组件。调整应用程序窗口的大小时，可以看到组件是如何立刻重新排列的。将窗口的宽度增大到原来的两倍，将发现所有**JButton**组件都显示在同一行。

14.1.1 **GridLayout**管理器

GridLayout类位于**java.awt**包中，它将容器中所有的组件组织为指定的行数和列数。分配给每个组件的显示区域都相同，因此如果指定3行3列的网格，容器将被划分成9个大小相等的区域。

当组件加入到容器中时，**GridLayout**将所有的组件放置到网格中的某个位置，而且组件是从左向右依次添加，当这一行满了之后，在从下一行的最左边开始添加。

下面的语句创建了一个容器，并将其设置为使用网格布局，该网格为2行3列：

```
GridLayout grid = new GridLayout(2, 3);  
setLayout(grid);
```

图14.2显示了使用网格布局时，应用程序Crisis的外观。

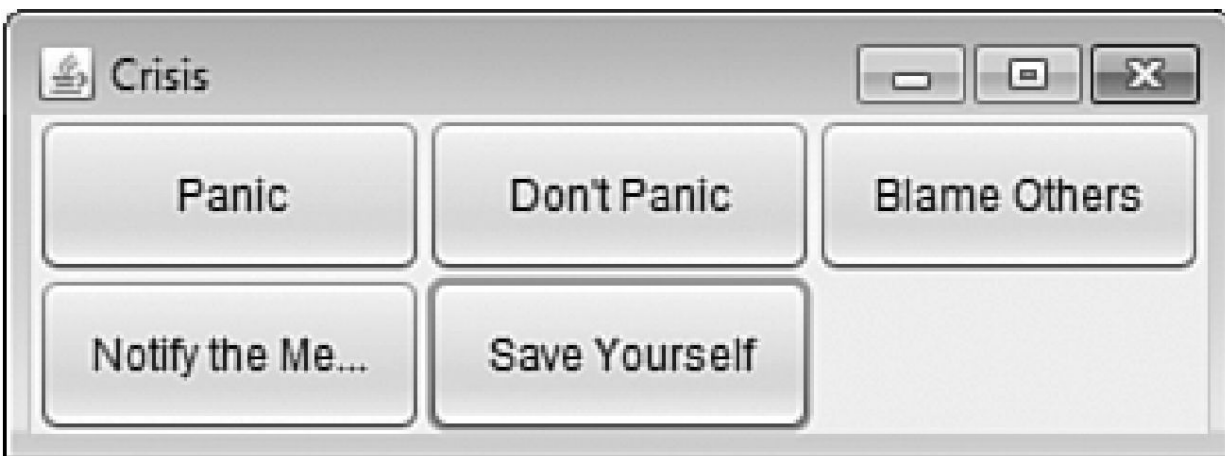


图14.2 使用网格布局排列组件

在图14.2中，有些标签显示的文本被截短。如果组件容纳不下标签文本，将使用省略号 (...) 表示省略的文本。

By the Way

注意

本书的配套网站中提供了Crisis程序的修改版本，它演示了本章介绍的每一种布局管理器（从CrisisGridDemo.java到网格布局）。要查看其源代

码，请访问www.java24hours.com，单击该书封面，然后找到第14章的页面。

14.1.2 BorderLayout管理器

BorderLayout类也位于**java.awt**包中，它将容器中的组件放置在特定的位置，该位置有5个方位：东、西、南、北、中。

BorderLayout管理器将组件放置到5个位置：其中4个位置由罗盘方向（compass direction）指定，另外一个由中心区域指定。当在该布局下添加组件时，**add()**方法会包含第2个参数，用于指示组件应该放置的位置。该参数应该是**BorderLayout**类的5个类变量之一：**NORTH**、**SOUTH**、**EAST**、**WEST**和**CENTER**。

与**GridLayout**类相同，**BorderLayout**也会将所有可用空间都分配给组件。在周围放置4个边界组件后，余下的空间都分配给中央的组件，因此它通常是最大的。

下面的语句创建一个使用边界布局的容器：

```
BorderLayout crisisLayout = new BorderLayout();
setLayout(crisisLayout);
add(panicButton, BorderLayout.NORTH);
add(dontPanicButton, BorderLayout.SOUTH);
add(blameButton, BorderLayout.EAST);
add(mediaButton, BorderLayout.WEST);
add(saveButton, BorderLayout.CENTER);
```

图14.3是使用这种布局时应用程序Crisis的外观。

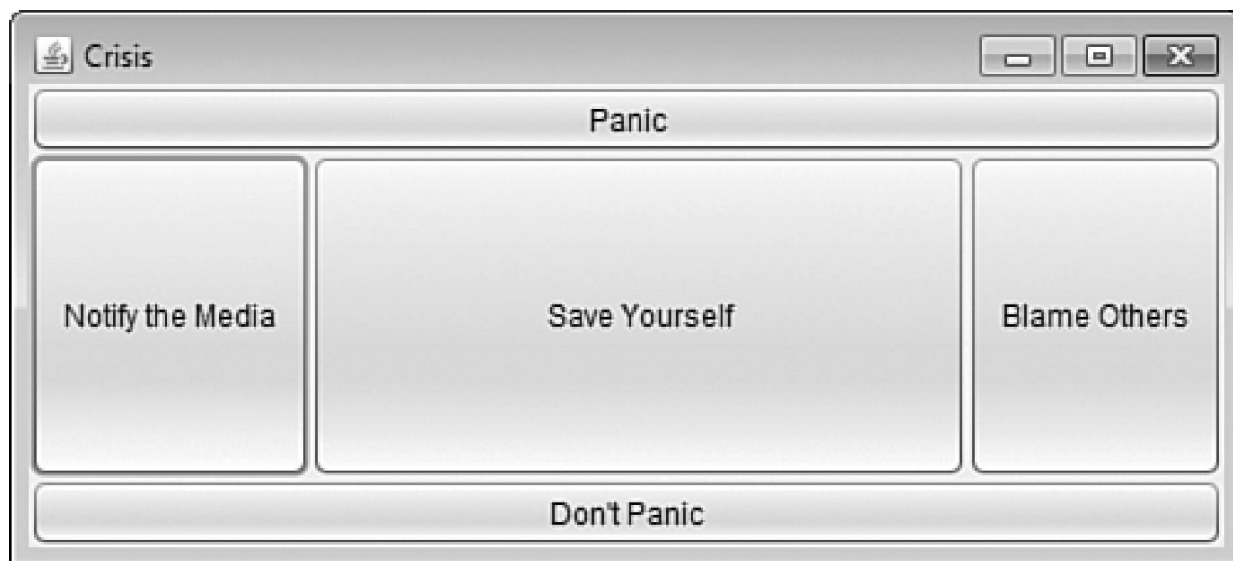


图14.3 使用边界布局排列组件

14.1.3 BorderLayout管理器

另一种方便的布局管理器是BoxLayout，它位于javax.swing包中，它可以将组件排列成一行或一列。

使用该布局时，先创建一个放置组件的面板，然后再创建一个布局管理器，它带有2个参数：

- 以框式布局（box layout）组织的组件；
- BoxLayout.Y_AXIS指定垂直排列，BoxLayout.X_AXIS指定水平排列。

下面是用于在应用程序Crisis中排列组件的代码：

```
JPanel pane = new JPanel();
BoxLayout box = new BoxLayout(pane, BoxLayout.Y_AXIS);
pane.setLayout(box);
pane.add(panicButton);
pane.add(dontPanicButton);
```

```
pane.add(blameButton);  
pane.add(mediaButton);  
pane.add(saveButton);  
add(pane);
```

图14.4显示了组件的排列情况。



图14.4 使用框式布局排列组件

14.1.4 使用Insets将组件隔开

在容器中排列组件时，可以使用Insets令组件远离容器边缘。Insets是代表容器边界区域的对象。

Insets类位于java.awt包中，它有一个接受4个参数的构造函数：在容器上、下、左、右留出的空间。每个参数都以像素为单位，像素是定义框架大小时使用的度量单位。

下面的语句创建一个Insets对象：

```
Insets around = new Insets(10, 6, 10, 3);
```

around对象代表容器的边界：上边缘内10像素、左边缘内6像素、下边缘内10像素、右边缘内3像素。

要想在容器中使用Insets对象，必须覆盖容器的getInsets()方法。该方法不接受任何参数，并返回一个Insets对象，如下所示：

```
public Insets getInsets() {  
    Insets squeeze = new Insets(50, 15, 10, 15);  
    return squeeze;  
}
```

图14.5说明了这对图14.1的影响，后者使用BorderLayout布局管理器。



图14.5 使用Insets增大组件周围的空间

在图14.5所示的容器中，左边框为15像素，下边框为10像素，右边框为15像素，上边框为50像素。

By the Way

注意

JFrame容器有内置的Inset，用于为标题栏留出空间。当覆盖getInsets()方法时并设置自定义的值时，过小的inset将导致容器将组件显示在标题栏下面。

14.2 应用程序的界面布局

到目前为止，布局管理器应用于整个框架：调用框架的setLayout()方法，所有组件遵循相同的规则。这适用于有些程序，但使用Swing开发图形用户界面时，将经常发现这些布局管理器都不合适。

解决这种问题的方法之一是，将一组JPanel对象作为容器，用于放置图形用户界面的不同部分。对于其中每部分，可以使用JPanel对象的setLayout()方法设置不同的布局规则。这些面板包含需要包含的组件后，就可以将这些面板直接加入到框架中。

接下来将开发一个完整的界面，该界面可以用于你在下一章编写的程序。该程序是一个猜数游戏，确定用户一生中赢得数百万大奖的机会。它不断随机生成6个数，直到用户选择的数字与摇出的数字相同。图14.6是要开发的应用程序GUI。

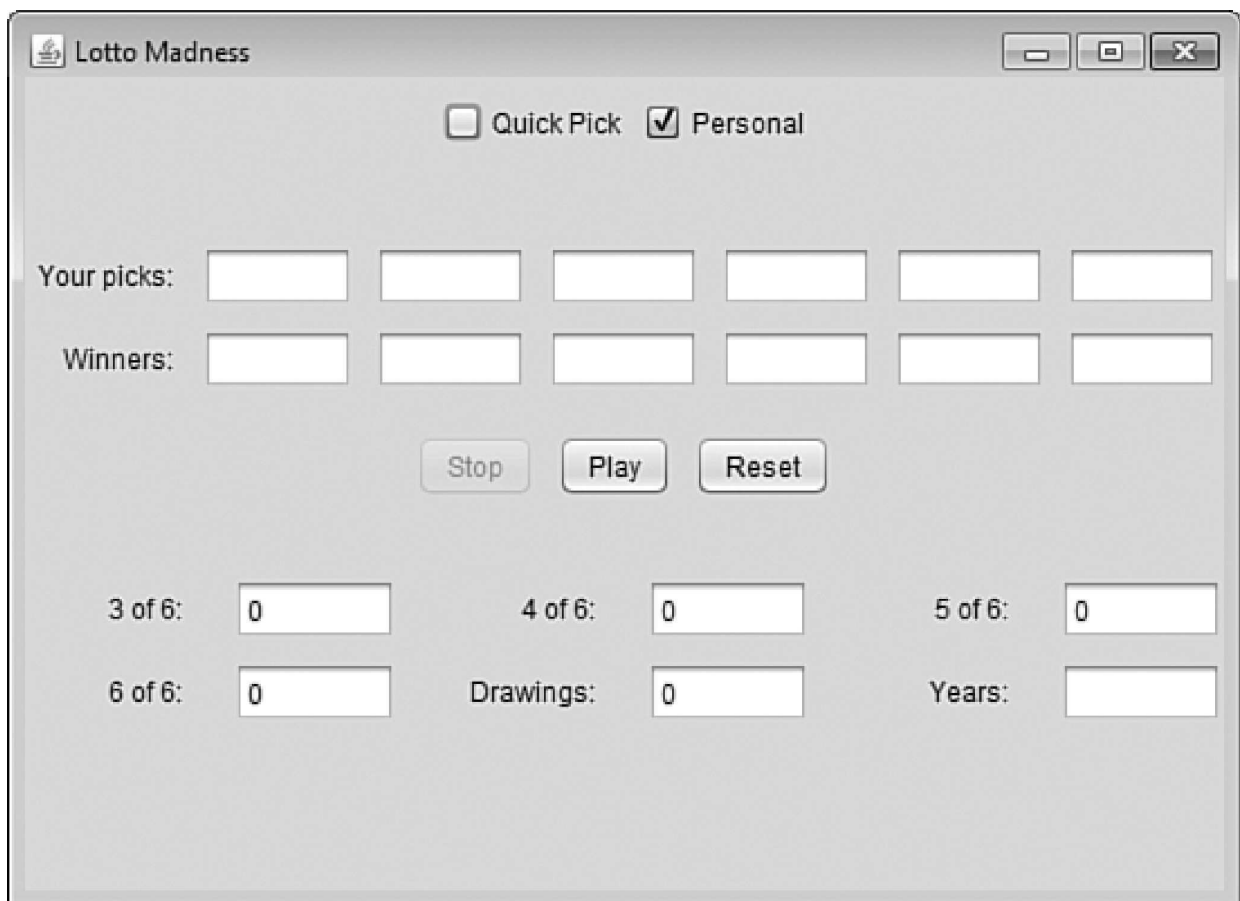


图14.6 显示应用程序LottoMadness的图形用户界面

创建一个新的Java空文件，将其命名为LottoMadness，然后输入程序清单14.2中的所有文本，并保存。

程序清单14.2 LottoMadness.java的完整源代码

```
1: package com.java24hours;
2:
3: import java.awt.*;
4: import javax.swing.*;
5:
6: public class LottoMadness extends JFrame {
7:
8:     // set up row 1
9:     JPanel row1 = new JPanel();
10:    ButtonGroup option = new ButtonGroup();
11:    JCheckBox quickpick = new JCheckBox("Quick Pick", false);
12:    JCheckBox personal = new JCheckBox("Personal", true);
13:    // set up row 2
14:    JPanel row2 = new JPanel();
15:    JLabel numbersLabel = new JLabel("Your picks: ",
JLabel.RIGHT);
16:    JTextField[] numbers = new JTextField[6];
17:    JLabel winnersLabel = new JLabel("Winners: ",
JLabel.RIGHT);
18:    JTextField[] winners = new JTextField[6];
19:    // set up row 3
20:    JPanel row3 = new JPanel();
21:    JButton stop = new JButton("Stop");
22:    JButton play = new JButton("Play");
23:    JButton reset = new JButton("Reset");
24:    // set up row 4
25:    JPanel row4 = new JPanel();
26:    JLabel got3Label = new JLabel("3 of 6: ", JLabel.RIGHT);
27:    JTextField got3 = new JTextField("0");
28:    JLabel got4Label = new JLabel("4 of 6: ", JLabel.RIGHT);
29:    JTextField got4 = new JTextField("0");
30:    JLabel got5Label = new JLabel("5 of 6: ", JLabel.RIGHT);
31:    JTextField got5 = new JTextField("0");
32:    JLabel got6Label = new JLabel("6 of 6: ", JLabel.RIGHT);
33:    JTextField got6 = new JTextField("0", 10);
34:    JLabel drawingsLabel = new JLabel("Drawings: ",
JLabel.RIGHT);
35:    JTextField drawings = new JTextField("0");
36:    JLabel yearsLabel = new JLabel("Years: ", JLabel.RIGHT);
37:    JTextField years = new JTextField();
38:
```

```
39: public LottoMadness() {
40:     super("Lotto Madness");
41:
42:     setSize(550, 400);
43:     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
44:     GridLayout layout = new GridLayout(5, 1, 10, 10);
45:     setLayout(layout);
46:
47:     FlowLayout layout1 = new FlowLayout(FlowLayout.CENTER,
48:         10, 10);
49:     option.add(quickpick);
50:     option.add(personal);
51:     row1.setLayout(layout1);
52:     row1.add(quickpick);
53:     row1.add(personal);
54:     add(row1);
55:
56:     GridLayout layout2 = new GridLayout(2, 7, 10, 10);
57:     row2.setLayout(layout2);
58:     row2.add(numbersLabel);
59:     for (int i = 0; i < 6; i++) {
60:         numbers[i] = new JTextField();
61:         row2.add(numbers[i]);
62:     }
63:     row2.add(winnersLabel);
64:     for (int i = 0; i < 6; i++) {
65:         winners[i] = new JTextField();
66:         winners[i].setEditable(false);
67:         row2.add(winners[i]);
68:     }
69:     add(row2);
70:
71:     FlowLayout layout3 = new FlowLayout(FlowLayout.CENTER,
72:         10, 10);
73:     row3.setLayout(layout3);
74:     stop.setEnabled(false);
75:     row3.add(stop);
76:     row3.add(play);
77:     row3.add(reset);
78:     add(row3);
79:
80:     GridLayout layout4 = new GridLayout(2, 3, 20, 10);
81:     row4.setLayout(layout4);
82:     row4.add(got3Label);
83:     got3.setEditable(false);
84:     row4.add(got3);
85:     row4.add(got4Label);
86:     got4.setEditable(false);
87:     row4.add(got4);
88:     row4.add(got5Label);
89:     got5.setEditable(false);
```

```

90:         row4.add(got5);
91:         row4.add(got6Label);
92:         got6.setEditable(false);
93:         row4.add(got6);
94:         row4.add(drawingsLabel);
95:         drawings.setEditable(false);
96:         row4.add(drawings);
97:         row4.add(yearsLabel);
98:         years.setEditable(false);
99:         row4.add(years);
100:        add(row4);
101:
102:        setVisible(true);
103:    }
104:
105:    private static void setLookAndFeel() {
106:        try {
107:            UIManager.setLookAndFeel(
108:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
109:            );
110:        } catch (Exception exc) {
111:            // ignore error
112:        }
113:    }
114:
115:    public static void main(String[] arguments) {
116:        LottoMadness.setLookAndFeel();
117:        LottoMadness frame = new LottoMadness();
118:    }
119: }

```

尽管现在还没有在该程序中添加任何语句，让其执行某些工作，但是现在也可以运行该程序，来确定图形界面的布置是否正确，是否可以收集所需要的信息。

该应用程序使用了几种不同的布局管理器。为更清楚应用程序用户界面的布局，请看图14.7。该界面分成4行，各排间用水平黑线分

隔。每行都是一个JPanel对象，应用程序使用整体布局管理器将这些行组织成4行1列的网格布局。

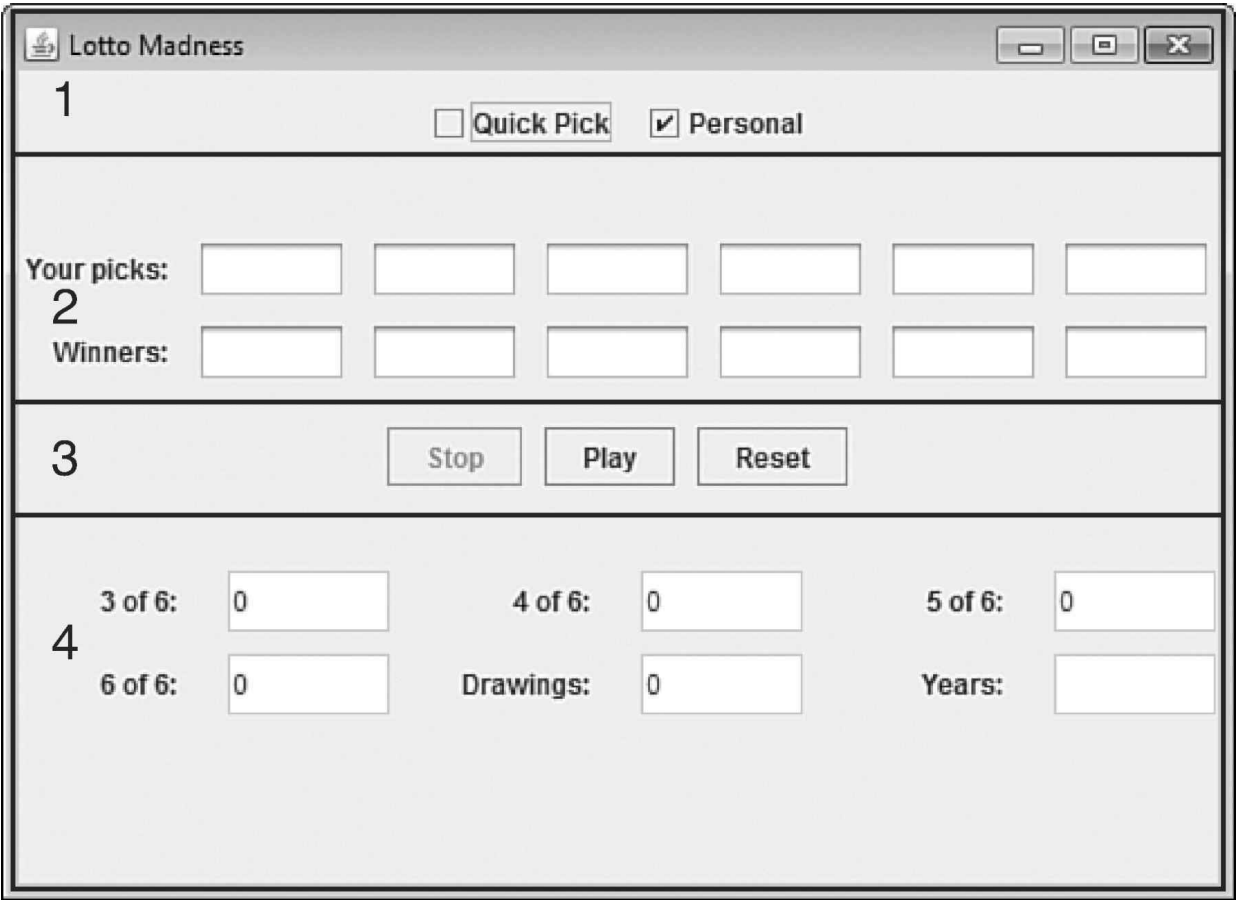


图14.7 将LottoMadness应用程序划分为不同的面板

在各行中，使用不同的布局管理器来确定组件的显示方式，第1行和第3行使用FlowLayout对象，程序的第47~48行说明这些对象如何创建的：

```
FlowLayout layout1 = new FlowLayout(FlowLayout.CENTER,
    10, 10);
```

在FlowLayout()构造函数中使用了3个参数。第一个参数是FlowLayout.CENTER，指定组件应在容器（它们所属的水平JPanel）中居中。后面两个参数指定每一个组件距离其他组件的宽度和高度，这里的宽度和高度分别为10像素，使得组件之间有一个较小的额外间隔距离。

界面的第2行被设置为2行7列的网格。在GridLayout()构造函数中，也将组件之间的水平距离和垂直距离指定为10像素。程序的第56～57行设置了该网格：

```
GridLayout layout2 = new GridLayout(2, 7, 10, 10);  
row2.setLayout(layout2);
```

界面的第4行使用GridLayout将组件组织成2行3列的网格。

应用程序LottoMadness使用了本章介绍的多种组件。程序代码第9～37行创建了界面中所有的组件对象。语句是按照行进行组织的。首先创建用于当前行的JPanel对象，然后创建这排中的组件。这段代码创建了所有的组件和容器，但调用add()方法将它们加入到应用程序的主框架中之前，这些组件和容器不会显示出来。

第47～100行添加组件。第47～54行是构造函数LottoMadness()：

```
FlowLayout layout1 = new FlowLayout(FlowLayout.CENTER,  
    10, 10);  
option.add(quickpick);  
option.add(personal);  
row1.setLayout(layout1);
```



```
row1.add(quickpick);  
row1.add(personal);  
add(row1);
```

创建完布局管理器对象后，将其作为行的JPanel对象（这里为row1）的setLayout()方法的参数。指定布局后，使用 add()方法将组件加入到JPanel中。放置好所有的组件后，调用row1对象的add()方法，将row1对象加入到框架中。

LottoMadness应用程序的图形用户界面的外观与之前的Swing应用程序并不相同，setLookAndFeel()是作为类方法创建的（注意第105行中的static关键字），然后在第116行被main()方法调用。

之前的应用程序将setLookAndFeel()方法穿件为一个对象方法，并在对象的构造函数中调用。之所以不在LottoMadness中这样做，是因为在创建任何实例变量并为其赋值之前，必须先选择应用程序的外观。

14.3 总结

当第一次设计Java程序的图形用户界面时，可能不相信组件能够移动是优点。布局管理器提供了开发引人入胜的图形用户界面的途径，它足够灵活，能够应对不同的显示方式。

下一章将更详细地介绍图形用户界面的功能。你将会使用LottoMadness界面来摇号，确定中奖号码。

14.4 问与答

问：在LottoMadness应用程序中，为何有些文本框为灰色，而其他文本框为白色？

答：使用setEditable()方法将有些文本框设置为不能编辑的，因此是灰色的。文本框的默认行为是，允许用户在文本框内单击并输入来修改文本框的值。然而有些文本框只是用来显示信息而不是接收用户输入。setEditable()方法可防止用户更改不应修改的文本框。

14.5 测验

通过回答下列问题可以检测对Java布局管理器的理解程度。

14.5.1 问题

1. 当将界面分成几个使用不同布局管理器的部分，通常使用下面哪种容器？

- a.** JWindow。
- b.** JPanel。
- c.** Container。

2. 面板的默认布局管理器是什么？

- a.** FlowLayout。

- b. GridLayout。
 - c. 没有默认布局管理器。
- 3. BorderLayout类因什么而得名?
 - a. 每个组件的边界。
 - b. 组件沿容器边界的排列方式。
 - c. Java的开发者随便取的。

14.5.2 答案

- 1. b. JPanel是最简单的容器。
- 2. a. 面板默认使用流式布局，但框架和窗口默认使用边界布局。
- 3. b. 将组件加入到容器中时，必须使用BorderLayout.WEST和BorderLayout.EAST等方向变量指定其边界位置。

14.6 练习

如果你想继续深入了解流式（网格和边界）布局，请完成下面的练习。

- 修改应用程序Crisis，使用一种布局管理器来组织对象panic和dontPanic，使用另一种布局管理器来组织其他3个按钮。

- 复制文件LottoMadness.java，然后将其重命名为NewMadness.java。修改该程序，将quick pick和personal choice作为组合框，将按钮start、stop和reset改为复选框。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站www.java24hours.com。

第15章 响应用户输入

本章介绍如下内容：

- 让程序感知事件；
- 设置组件，使其引发事件；
- 忽略一些事件；
- 找出程序中事件的结束点；
- 在界面中存储信息；
- 准换存储在文本框中的值。

在前两章开发的图形用户界面无须任何修改就能独立运行。用户可以单击按钮，在文本框输入文本，调整窗口大小。然而，即使最没有识别能力的用户迟早也会提出更多的需求，因此程序提供的图形用户界面必须响应鼠标单击和键盘输入。

如果Java程序能够响应用户事件，就可以实现上述功能。响应用户事件通常称为事件处理，这是本章要介绍的主题。

15.1 让程序监听

在Java中，用户事件是这样定义的：当用户使用鼠标、键盘或其他输入设备执行某种操作时，所引发的行为。

在接收事件之前，必须知道如何让对象监听。要响应用户事件，必须使用一个或多个**EventListener**接口。接口是Java面向对象编程的一个特性，能够让类继承原本无法使用的行为。接口很像与其他类签订的合同，用来确保类包含具体的方法。

EventListener接口包含的方法可以接受特定类型的用户输入信息。

要添加**EventListener**接口，必须完成两项工作。首先，因为监听类位于**java.awt.event**包中，因此必须通过下面的语句使其在程序中可用：

```
import java.awt.event.*;
```

其次，必须使用关键字**implements**将类声明为支持一个或多个监听接口。下面的语句创建一个这样的类，即使用**ActionListener**接口来响应按钮和菜单单击：

```
public class Graph implements ActionListener {
```

EventListener接口让图形用户界面中的组件生成用户事件。如果一个监听器都没有，组件将不能做任何让程序的其他部分能够知道的事情。程序中每个要监听的组件都必须有监听器接口。要让程序响应用鼠标单击按钮或在文本框中按回车键，必须包括**ActionListener**接口。要对使用选择列表或复选框进行响应，需要使用**ItemListener**接口。

在同一个类中需要多个接口时，可在关键字**implements**的后面列出接口名，并用逗号将它们隔开，如下列代码所示：

```
public class Graph3D implements ActionListener, MouseListener {  
    // ...  
}
```

15.2 设置要监听的组件

为组件实现所需的接口后，还必须设置该组件使其生成用户事件。**ActionListener**接口监听操作事件，比如单击按钮或按下回车键。

要让 **JButton**对象生成一个事件，可使用**addActionListener()**方法，如下所示：

```
JButton fireTorpedos = new JButton("Fire torpedos");  
fireTorpedos.addActionListener(this);
```

这段代码创建按钮**fireTorpedos**，然后调用其**addActionListener()**方法。通过将**this**关键字作为参数传递给**addActionListener()**方法，指出当前对象将接收用户事件并在需要时进行处理。

By the
Way

注意

很多读者在第一次接触到**this**关键字时，并不懂它的含义。关键字**this**指的是其所在的对象。因此，如果创建一个**LottoMadess**类，并在一条语句中使用**this**，则它指的是执行该语句的**LottoMadess**对象。

15.3 处理用户事件

当有监听器的组件生成一个用户事件时，将自动调用一个方法，该方法位于将监听器同组件关联起来时指定的类中。

每个监听器有不同的方法，用于接收其事件。**ActionListener**接口将事件发送给方法**action Performed()**。下面是一个简短的**actionPerformed()**方法示例：

```
public void actionPerformed(ActionEvent event) {  
    // method goes here  
}
```

程序中所有的操作事件都将发送给该方法。如果程序只有一个组件可以发送操作事件，可以将处理事件的语句放在该方法中。如果程序有多个组件可以发送操作事件，则需要检查发送到方法的对象。

一个**ActionEvent**对象被发送到**actionPerformed()**方法。有几种对象用于表示可以在程序中发送的用户事件。这些类包含可用于判断事件

由哪个组件引发的方法。在`actionPerformed()`方法中，如果`ActionEvent`对象名为`event`，可使用下面的语句来确定引发事件的组件：

```
String cmd = event.getActionCommand();
```

`getActionCommand()`方法返回一个字符串。如果引发事件的组件是按钮，返回的字符串将是该按钮的标签；如果是文本框，返回的字符串为文本框包含的文本。`getSource()`方法返回引发事件的对象。

可以使用下面的`actionPerformed()`方法接收来自3个组件的事件：名为`start`的`JButton`对象、名为`speed`的`JTextField`对象和名为`viscosity`的`JTextField`对象：

```
public void actionPerformed(ActionEvent event) {  
    Object source = event.getSource();  
    if (source == speed) {  
        // speed field caused event  
    } else if (source == viscosity) {  
        // viscosity caused event  
    } else {  
        // start caused event  
    }  
}
```

可以在所有用户事件上使用`getSource()`方法，以确定引发任何事件的具体对象。

15.3.1 复选框和组合框事件

复选框和组合框需要ItemListener接口。调用组件的addItemListener()方法使其生成用户事件。例如，下面的语句创建一个名为superSize的复选框，并使其被选中或取消选中时发送用户事件：

```
JCheckBox superSize = new JCheckBox("Super Size", true);
superSize.addItemListener(this);
```

这些事件由itemStateChanged()方法接收，该方法将一个ItemEvent对象作为参数。要确定事件是由哪个对象引发的，可调用事件对象的getItem()方法。

要确定复选框是否被选中，可将方法getStateChange()方法返回的值与常量ItemEvent.SELECTED、ItemEvent.DESELECTED进行比较。下面代码是一个名为item的ItemEvent对象示例：

```
public void itemStateChanged(ItemEvent item) {
    int status = item.getStateChange();
    if (status == ItemEvent.SELECTED) {
        // item was selected
    }
}
```

要确定JComboBox对象中选定的值，可使用getItem()方法并将返回值转换为字符串，如下所示：

```
Object which = item.getItem();
```

```
String answer = which.toString();
```

15.3.2 键盘事件

当一个键按下，程序需要立即做出响应时，它使用的是键盘事件和KeyListener接口。

第一步是调用组件的addKeyListener()方法，注册接收按键事件的组件。该方法的参数应为实现KeyListener接口的对象，如果是当前类，可使用this作为参数。

处理键盘事件的对象必须实现以下3个方法。

- void keyPressed(KeyEvent): 按下键时调用该方法。
- void keyReleased(KeyEvent): 松开键时调用该方法。
- void keyTyped(KeyEvent): 按下并松开键时调用该方法。

其中每个方法都将一个KeyEvent对象作为参数，可以调用该对象的方法来更详细地了解事件。调用getKeyChar()方法可确定按下的是哪个键，这是通过返回一个char值来指出的，且只适用于字母、数字和标点符号。

要监视键盘上的每个键，包括回车键、Home键、Page Up键和Page Down键，可调用getKeyCode()方法。该方法返回一个代表键的整数，然后将该整数作为参数调用getKeyText()方法，从而返回一个包含键名（如Home、F1等）的String对象。

程序清单15.1所示的Java应用程序使用getKeyChar()方法在标签上显示最近按下的键。该应用程序实现了KeyListener接口，因此包含方法keyTyped()、keyPressed()和keyReleased()，但只在第24~27行实现了keyTyped()方法。创建一个名为KeyViewer的Java文件，然后输入程序清单15.1中的全部文本，并保存。

程序清单15.1 KeyView.java的完整源代码

```
1: package com.java24hours;
2:
3: import javax.swing.*;
4: import java.awt.event.*;
5: import java.awt.*;
6:
7: public class KeyViewer extends JFrame implements KeyListener {
8:     JTextField keyText = new JTextField(80);
9:     JLabel keyLabel = new JLabel("Press any key in the text
field.");
10:
11:     public KeyViewer() {
12:         super("KeyViewer");
13:         setLookAndFeel();
14:         setSize(350, 100);
15:         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16:         keyText.addKeyListener(this);
17:         BorderLayout bord = new BorderLayout();
18:         setLayout(bord);
19:         add(keyLabel, BorderLayout.NORTH);
20:         add(keyText, BorderLayout.CENTER);
21:         setVisible(true);
22:     }
23:
24:     public void keyTyped(KeyEvent input) {
25:         char key = input.getKeyChar();
26:         keyLabel.setText("You pressed " + key);
27:     }
28:
29:     public void keyPressed(KeyEvent txt) {
30:         // do nothing
31:     }
32:
33:     public void keyReleased(KeyEvent txt) {
34:         // do nothing
```

```

35:     }
36:
37:     private void setLookAndFeel() {
38:         try {
39:             UIManager.setLookAndFeel(
40: "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
41:             );
42:         } catch (Exception exc) {
43:             // ignore error
44:         }
45:     }
46:
47:     public static void main(String[] arguments) {
48:         KeyViewer frame = new KeyViewer();
49:     }
50: }

```

运行该程序时，其输出结果应该如图15.1所示。



图15.1 在程序中处理键盘事件

15.3.3 启用和禁用组件

读者可能在程序看到过与众不同的组件，它呈灰色。

灰色表示用户不能对组件执行任何操作，因为它被禁用。组件的启用和禁用是在组件的`setEnabled()`方法中实现的。该方法接收一个布

尔值作为参数，因此setEnabled (true)启用组件，而setEnabled(false)禁用组件。

下面的语句创建3个按钮（它们的标签分别是Previous、Next和Finish）并禁用第一个按钮：

```
JButton previousButton = new JButton("Previous");
JButton nextButton = new JButton("Next");
JButton finishButton = new JButton("Finish");
previousButton.setEnabled(false);
```

该方法可以有效地防止一个组件在不应该发送用户事件时而进行发送。例如，编写Java应用程序，通过文本框接收用户地址时，可能希望用户在提供街道地址、城市、州名以及邮政编码之前，禁用Save Address按钮。

15.4 完善图形应用程序

为了理解如何在Java程序中使用Swing的事件处理类，这里将完善第14章介绍过的摇奖模拟程序LottoMadness。

当前，LottoMadness只是一个图形用户界面。用户可以单击按钮或在文本框中输入文本，但没有任何响应。这里将创建一个LottoEvent类，它接收用户输入、进行摇奖并记录用户中奖的次数。编写好这个类后，需要在LottoMadness中添加几行代码，以便使用LottoEvent。通

常以这种方式划分Swing项目：将图形用户界面放在一个类中，将事件处理方法放在另一个类中。

该应用程序旨在评估用户一生中赢六合彩的几率。图15.2是该程序运行时的屏幕截图。



图15.2 运行LottoMadness 程序

这里没有使用概率论来解决这个问题，计算机使用了一种更有趣的解决方法：它不断摇奖，直到用户中奖。由于6个数全中几乎不可能，用户猜对其中的3个、4个和5个数时，程序也将报告。

在创建的界面中，包含用于显示六合彩号码的12文本框和两个标签分别为**Quick Pick**和**Personal**的复选框。其中有6个文本框是不能输入的，它们用于显示每次摇出的中奖号码；其他6个文本框显示用户选择的号码。当用户选中**Quick Pick**复选框时，将为用户随机显示6个数字；如果选中复选框**Personal**，则用户可以选择所需要的数字。

3个按钮用于控制程序：**Stop**、**Play**和**Reset**。用户单击按钮**Play**时，程序将调用线程**playing**，并生成中奖号码。

用户按下**Stop**按钮时，线程停止，再按下**Reset**按钮将清除所有的文本框内容，以便开始新一轮游戏。第19章将详细讲解线程。

LottoEvent类实现了3个接口：**ActionListener**、**ItemListener**和**Runnable**。**Runnable**接口与线程相关。该程序需要一个接听者来监听由应用程序的按钮和复选框生成的用户事件，但是它不需要监听与文本框相关的任何事件，因为这些文本框只用于存储用户选择的数字。用户界面将自动处理这种存储功能。

这个类需要使用Swing主包**javax.swing**和Java事件处理包**java.awt.event**。

这个类包含如下两个实例变量。

- **gui**: 一个**LottoMadness**对象。
- **playing**: 用于不断摇奖的**Thread**对象。

变量**gui**用于同包含图形用户界面的**LottoMadness**对象通信。当需要修改界面或检索文本框中的值时，需要使用**gui**对象的实例变量。

例如，LottoMadness的实例变量play代表Play按钮，要在LottoEvent中禁用该按钮，可使用下面的语句：

```
gui.play.setEnabled(false);
```

下面的语句可用于检索JTextField对象got3的值：

```
String got3value = gui.got3.getText();
```

程序清单15.2列出了LottoEvent类的全部源代码。在NetBeans中创建一个名为Lotto Event的Java空文件，然后输入程序清单15.2中的所有文本，并保存。

程序清单15.2 LottoEvent.java的完整源代码

```
1: package com.java24hours;
2:
3: import javax.swing.*;
4: import java.awt.event.*;
5:
6: public class LottoEvent implements ItemListener, ActionListener,
7:     Runnable {
8:
9:     LottoMadness gui;
10:    Thread playing;
11:
12:    public LottoEvent(LottoMadness in) {
13:        gui = in;
14:    }
15:
```

```
16:     public void actionPerformed(ActionEvent event) {
17:         String command = event.getActionCommand();
18:         if (command.equals("Play")) {
19:             startPlaying();
20:         }
21:         if (command.equals("Stop")) {
22:             stopPlaying();
23:         }
24:         if (command.equals("Reset")) {
25:             clearAllFields();
26:         }
27:     }
28:
29:     void startPlaying() {
30:         playing = new Thread(this);
31:         playing.start();
32:         gui.play.setEnabled(false);
33:         gui.stop.setEnabled(true);
34:         gui.reset.setEnabled(false);
35:         gui.quickpick.setEnabled(false);
36:         gui.personal.setEnabled(false);
37:     }
38:
39:     void stopPlaying() {
40:         gui.stop.setEnabled(false);
41:         gui.play.setEnabled(true);
42:         gui.reset.setEnabled(true);
43:         gui.quickpick.setEnabled(true);
44:         gui.personal.setEnabled(true);
45:         playing = null;
46:     }
47:
48:     void clearAllFields() {
49:         for (int i = 0; i < 6; i++) {
50:             gui.numbers[i].setText(null);
51:             gui.winners[i].setText(null);
52:         }
53:         gui.got3.setText("0");
54:         gui.got4.setText("0");
55:         gui.got5.setText("0");
56:         gui.got6.setText("0");
57:         gui.drawings.setText("0");
58:         gui.years.setText("0");
59:     }
60:
61:     public void itemStateChanged(ItemEvent event) {
62:         Object item = event.getItem();
63:         if (item == gui.quickpick) {
64:             for (int i = 0; i < 6; i++) {
65:                 int pick;
66:                 do {
```

```

67:             pick = (int) Math.floor(Math.random() *
50 + 1);
68:             } while (numberGone(pick, gui.numbers, i));
69:             gui.numbers[i].setText("" + pick);
70:         }
71:     } else {
72:         for (int i = 0; i < 6; i++) {
73:             gui.numbers[i].setText(null);
74:         }
75:     }
76: }
77:
78: void addOneToField(JTextField field) {
79:     int num = Integer.parseInt("0" + field.getText());
80:     num++;
81:     field.setText("" + num);
82: }
83:
84: boolean numberGone(int num, JTextField[] pastNums, int
count) {
85:     for (int i = 0; i < count; i++) {
86:         if (Integer.parseInt(pastNums[i].getText()) ==
num) {
87:             return true;
88:         }
89:     }
90:     return false;
91: }
92:
93: boolean matchedOne(JTextField win, JTextField[] allPicks) {
94:     for (int i = 0; i < 6; i++) {
95:         String winText = win.getText();
96:         if ( winText.equals( allPicks[i].getText() ) ) {
97:             return true;
98:         }
99:     }
100:     return false;
101: }
102:
103: public void run() {
104:     Thread thisThread = Thread.currentThread();
105:     while (playing == thisThread) {
106:         addOneToField(gui.drawings);
107:         int draw =
Integer.parseInt(gui.drawings.getText());
108:         float numYears = (float)draw / 104;
109:         gui.years.setText("" + numYears);
110:
111:         int matches = 0;
112:         for (int i = 0; i < 6; i++) {
113:             int ball;

```

```

114:                do {
115:                    ball = (int) Math.floor(Math.random()
* 50 + 1);
116:                } while (numberGone(ball, gui.winners, i));
117:                gui.winners[i].setText("" + ball);
118:                if (matchedOne(gui.winners[i],
gui.numbers)) {
119:                    matches++;
120:                }
121:            }
122:            switch (matches) {
123:                case 3:
124:                    addOneToField(gui.got3);
125:                    break;
126:                case 4:
127:                    addOneToField(gui.got4);
128:                    break;
129:                case 5:
130:                    addOneToField(gui.got5);
131:                    break;
132:                case 6:
133:                    addOneToField(gui.got6);
134:                    gui.stop.setEnabled(false);
135:                    gui.play.setEnabled(true);
136:                    playing = null;
137:            }
138:            try {
139:                Thread.sleep(100);
140:            } catch (InterruptedException e) {
141:                // do nothing
142:            }
143:        }
144:    }
145: }

```

LottoEvent类有一个构造函数: LottoEvent (LottoMadness) , 它将LottoMadness对象作为参数, 这表明要依靠LottoEvent来处理用户事件和进行摇奖。

在这个类中使用下述方法来完成任务。

- `clearAllField()`方法清空应用程序中所有的文本框，它在用户单击Reset按钮时被调用。
- `addOneToField()`方法将文本框的内容转换为整数值，将其加1后再转换为字符串并存储到文本框中。由于所有的文本框存储的都是字符串，要将其作为数字使用，必须采取特殊措施。
- `numberGone()`方法接收3个参数—已摇出的一个数字、可存储多个JTextField对象的数组和一个count整数。该方法确保中奖号码不包含相同的数字。
- `matchedOne()`方法接收2个参数—一个JTextField对象和一个包含6个JTextField对象的数组。该方法检查用户选中的某个数是否与当前中奖号码中的数字相同。

应用程序的`actionPerformed()`方法接收用户单击按钮时引发的事件。`getActionCommand()`方法检索按钮的标签，以确定哪个按钮被单击。

单击Play按钮将调用`startPlaying()`方法，该方法禁用4个组件。单击Stop按钮将调用`stopPlaying()`方法，该方法启用除Stop按钮外的所有组件。

`ItemStateChanged()`方法接受用户选择复选框Quick Pick和Personal时触发的事件。`getItem()`方法返回一个对象，指出哪个复选框被单击。如果是复选框Quick Pick，则生成6个1~50的随机数，作为用户选择的彩票号码；否则清空用户用来输入号码的文本框。

LottoEvent类使用1~50的数字表示摇出的小球。这是在第115行完成的，它将方法`Math.random()`返回的值乘以50再加1，并将结果作为

`Math.floor()`方法的参数，最终得到一个1~50的随机整数。如果将这里和第67行的50替换为其他数字，可将LottoMadness用于进行更大或更小范围的摇奖模拟。

LottoMadness项目没有使用变量来记录摇出的号码、中奖次数和文本框中的彩票号码。相反，它使用界面来存储值并自动显示它们。

为完成该项目，在NetBeans中重新打开LottoMadness.java文件。只需要添加6行代码就可以使用LottoEvent类。

首先使用下面的语句添加一个新的实例变量，用于存储LottoEvent对象：

```
LottoEvent lotto = new LottoEvent(this);
```

接下来，在构造函数LottoMadness()中，对于每个可接收用户输入的界面组件，调用其方法addItemListener()和addActionListener()：

```
// Add listeners
quickpick.addItemListener(lotto);
personal.addItemListener(lotto);
stop.addActionListener(lotto);
play.addActionListener(lotto);
reset.addActionListener(lotto);
```

这些语句应该紧贴在构造函数的后面进行添加，并且位于显示组件的setVisible(true)语句之前。

程序清单15.3列出了修改之后的LottoMadness.java的完整源代码。其中新增的代码用黑色阴影表示，其他代码与前一章相同。

程序清单15.3 LottoMadnes.java的完整源代码

```
1: package com.java24hours;
2:
3: import java.awt.*;
4: import javax.swing.*;
5:
6: public class LottoMadness extends JFrame {
7:     LottoEvent lotto = new LottoEvent(this);
8:
9:     // set up row 1
10:    JPanel row1 = new JPanel();
11:    ButtonGroup option = new ButtonGroup();
12:    JCheckBox quickpick = new JCheckBox("Quick Pick", false);
13:    JCheckBox personal = new JCheckBox("Personal", true);
14:    // set up row 2
15:    JPanel row2 = new JPanel();
16:    JLabel numbersLabel = new JLabel("Your picks: ",
JLabel.RIGHT);
17:    JTextField[] numbers = new JTextField[6];
18:    JLabel winnersLabel = new JLabel("Winners: ",
JLabel.RIGHT);
19:    JTextField[] winners = new JTextField[6];
20:    // set up row 3
21:    JPanel row3 = new JPanel();
22:    JButton stop = new JButton("Stop");
23:    JButton play = new JButton("Play");
24:    JButton reset = new JButton("Reset");
25:    // set up row 4
26:    JPanel row4 = new JPanel();
27:    JLabel got3Label = new JLabel("3 of 6: ", JLabel.RIGHT);
28:    JTextField got3 = new JTextField("0");
29:    JLabel got4Label = new JLabel("4 of 6: ", JLabel.RIGHT);
30:    JTextField got4 = new JTextField("0");
31:    JLabel got5Label = new JLabel("5 of 6: ", JLabel.RIGHT);
32:    JTextField got5 = new JTextField("0");
33:    JLabel got6Label = new JLabel("6 of 6: ", JLabel.RIGHT);
34:    JTextField got6 = new JTextField("0", 10);
```

```
35:     JLabel drawingsLabel = new JLabel("Drawings: ",
JLabel.RIGHT);
36:     JTextField drawings = new JTextField("0");
37:     JLabel yearsLabel = new JLabel("Years: ", JLabel.RIGHT);
38:     JTextField years = new JTextField("0");
39:
40:     public LottoMadness() {
41:         super("Lotto Madness");
42:
43:         setSize(550, 400);
44:         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
45:         GridLayout layout = new GridLayout(5, 1, 10, 10);
46:         setLayout(layout);
47:
48:         // Add listeners
49:         quickpick.addItemListener(lotto);
50:         personal.addItemListener(lotto);
51:         stop.addActionListener(lotto);
52:         play.addActionListener(lotto);
53:         reset.addActionListener(lotto);
54:
55:         FlowLayout layout1 = new FlowLayout(FlowLayout.CENTER,
56:             10, 10);
57:         option.add(quickpick);
58:         option.add(personal);
59:         row1.setLayout(layout1);
60:         row1.add(quickpick);
61:         row1.add(personal);
62:         add(row1);
63:
64:         GridLayout layout2 = new GridLayout(2, 7, 10, 10);
65:         row2.setLayout(layout2);
66:         row2.add(numbersLabel);
67:         for (int i = 0; i < 6; i++) {
68:             numbers[i] = new JTextField();
69:             row2.add(numbers[i]);
70:         }
71:         row2.add(winnersLabel);
72:         for (int i = 0; i < 6; i++) {
73:             winners[i] = new JTextField();
74:             winners[i].setEditable(false);
75:             row2.add(winners[i]);
76:         }
77:         add(row2);
78:
79:         FlowLayout layout3 = new FlowLayout(FlowLayout.CENTER,
80:             10, 10);
81:         row3.setLayout(layout3);
82:         stop.setEnabled(false);
83:         row3.add(stop);
84:         row3.add(play);
```



```
85:         row3.add(reset);
86:         add(row3);
87:
88:         GridLayout layout4 = new GridLayout(2, 3, 20, 10);
89:         row4.setLayout(layout4);
90:         row4.add(got3Label);
91:         got3.setEditable(false);
92:         row4.add(got3);
93:         row4.add(got4Label);
94:         got4.setEditable(false);
95:         row4.add(got4);
96:         row4.add(got5Label);
97:         got5.setEditable(false);
98:         row4.add(got5);
99:         row4.add(got6Label);
100:        got6.setEditable(false);
101:        row4.add(got6);
102:        row4.add(drawingsLabel);
103:        drawings.setEditable(false);
104:        row4.add(drawings);
105:        row4.add(yearsLabel);
106:        years.setEditable(false);
107:        row4.add(years);
108:        add(row4);
109:
110:        setVisible(true);
111:    }
112:
113:    private static void setLookAndFeel() {
114:        try {
115:            UIManager.setLookAndFeel(
116:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
117:            );
118:        } catch (Exception exc) {
119:            // ignore error
120:        }
121:    }
122:
123:    public static void main(String[] arguments) {
124:        LottoMadness.setLookAndFeel();
125:        LottoMadness frame = new LottoMadness();
126:    }
127: }
```

在添加完阴影显示的代码行之后，可以运行该程序，它可以测试你玩彩票的技能。正如读者预期的，这种练习毫无意义，因为在一生中赢得六合彩的机会非常小，即使你活得和圣经中的人物一样长。

Did you Know?

提示

本书的配套网站www.java24hours.com包含了applet版本LottoMadness程序的链接。本书付印时，摇了410732244次奖，这相当于39000000年（每两周摇一次）。其中6个数字中猜对3个的有6364880次，猜对4个有337285次，猜对5个的有6476次，全部猜对的有51次。Bill Teer是第一个中了该虚拟六合彩的人，他的中奖号码是3、7、1、15、34和43，是在下了241225注（相当于2319.47年）后中的。

15.5 总结

通过使用Swing，只需进行少量的编程就可以创建具备专业级外观的程序。虽然应用程序LottoMadness比前14章创建的很多示例程序都长，但有一半语句是用于创建界面的。

如果花些时间运行该程序，将更加羡慕六合彩得主的好运气。

我最近运行该程序的结果表明，我挥霍27000美元和人生中美好的266年买奖票，却只有几注是6中4和6中3的。与这样的几率相比，通过选择Java编程技能来赚钱无疑更现实。

15.6 问与答

问：为反映文本框变化，需要使用方法`paint()`或`repaint()`吗？

答：使用文本组件的`setText()`方法修改其值后，无需再做其他事情。`Swing`负责处理必要的更新，以显示新的值。

问：为何经常同时导入一个包及其子包，例如，在程序清单15.1中导入`java.awt.*`和`java.awt.event.*`。难道第一条语句没有包含第二条吗？

答：虽然从名称上看，`java.awt`和`java.awt.event`是相关的，但在Java中包是不能继承的。一个包不可能是另一个包的子包。

在`import`语句中使用星号实际上是将包中所有的类导入。

星号只适用于类而不适用于包，因此单条`import`语句最多只能导入单个包中的所有类。

15.7 测验

如果 `LottoMadness` 程序激起了你玩游戏的热情，请回答下列问题以检测玩游戏的技能。

15.7.1 问题

1. 操作事件因何而得名？
 - a. 它们因响应其他事情而发生。
 - b. 表示为了进行响应而采取某种操作。

- c. 对电影中的冒险家Action Jackson表示敬意。
- 2. 作为方法addActionListener()的参数时，this表示什么？
 - a. 有事件发生时，应使用该监听器。
 - b. 该事件优先于其他事件。
 - c. 该对象将处理事件。
- 3. 下面哪个组件将用户输入作为整数存储？
 - a. JButton。
 - b. JTextArea。
 - c. 都不是。

15.7.2 答案

- 1. b. 操作事件包括单击按钮和从下拉菜单中选择一个菜单项。
- 2. c. 关键字this表示当前对象。如果使用的是对象名而不是关键字this，该对象将接收并处理事件。
- 3. c. JTextField和JTextArea组件都将其值存储为文本，因此要将其值作为整数、浮点数或其他非文本值使用时，必须进行转换。

15.8 练习

如果本章的主要事件没有满足你的胃口，请完成下列练习：

- 在应用程序LottoMadness中添加一个文本框，它与LottoEvent类的Thread.sleep()语句协同工作，以降低摇奖速度；
- 修改LottoMadness项目，使其摇出5个1～90的数字。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第16章 创建复杂的用户界面

本章介绍如下内容：

- 使用滑块输入数值；
- 监视用户通过滑块进行的输入；
- 创建图像图标和工具栏；
- 在表格中显示数据。

使用Swing创建图形用户界面时，除学习如何使用不同的界面组件、布局管理器和事件处理方法外，还必须熟悉Swing提供的所有功能。

Swing包含400多个不同的类，这使其成为Java中的一个最广泛的类库。其中的很多类可用前三章介绍的技术来实现——所有Swing容器和组件有相同的超类，这给了它们相同的行为。

本章将介绍可在Swing程序中使用的其他组件。

16.1 滑块

用来收集用户输入的数字的最简单方法之一是使用滑块，这是一个可上下或左右拖曳的组件。在Swing中，滑块用JSlider类表示。

滑块用于从一个最大值和最小值之间选择一个数字，这些值可显示在标签上，包括最小值、最大值和中间值（如图16.2所示）。后续将

要创建的一个示例如图16.1所示。

可使用下面的构造函数之一创建水平滑块。

- `JSlider()`: 创建最小值为0，最大值为100，初始值为50的滑块。
- `JSlider(int, int)`: 创建具有指定最小值和最大值的滑块。
- `JSlider(int, int, int)`: 创建具有指定最小值、最大值和初始值的滑块。

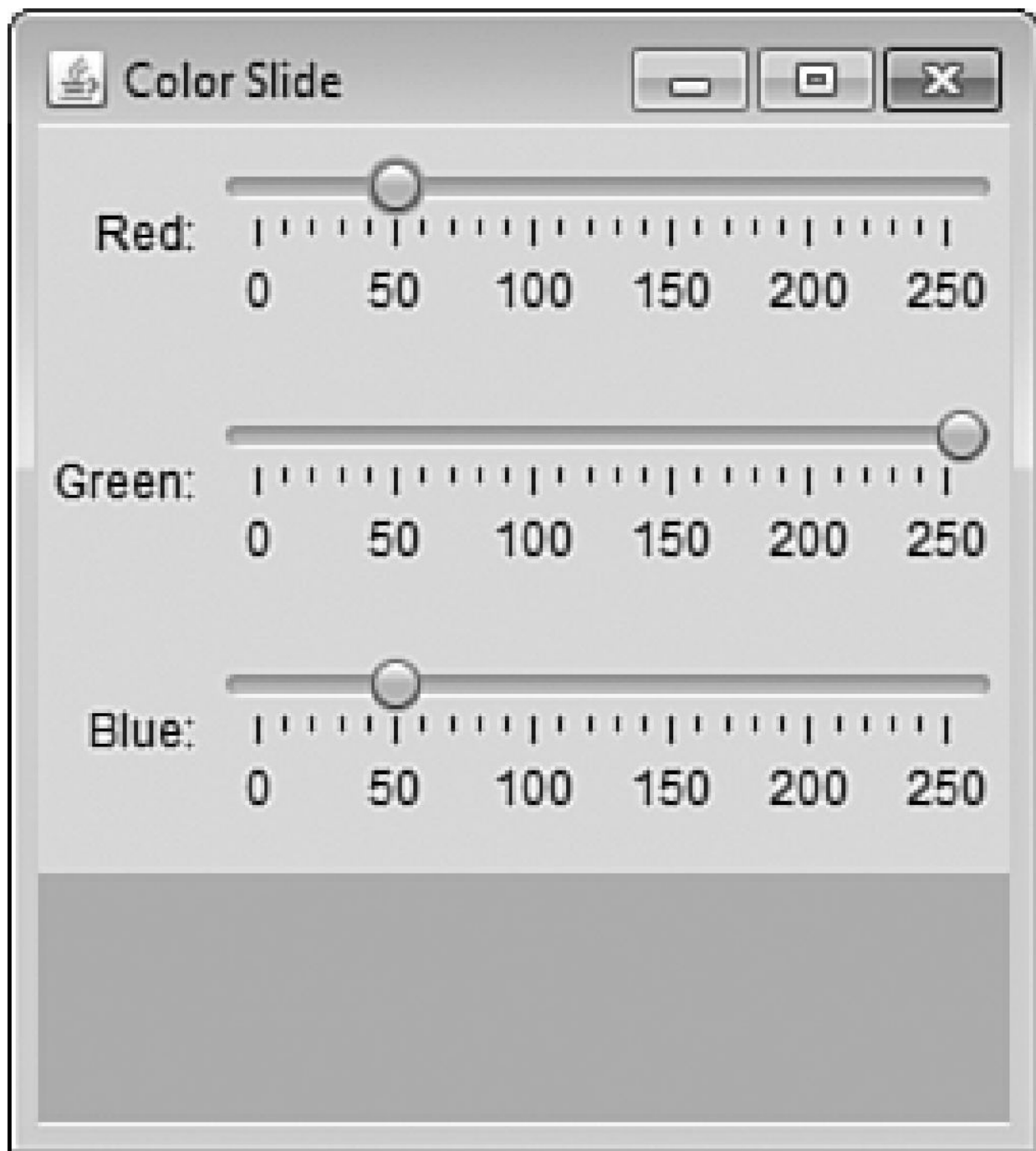


图16.1 使用3个滑块组件选择颜色

要创建垂直滑块，必须使用接受另外一个附加参数（即滑块方向）的构造函数，该参数应设置为类变量`Jslider.VERTICAL`或`Jslider.HORIZONTAL`。

下面的语句创建一个垂直滑块，可用于选择1~1000的数字：

```
JSlider guess = new JSlider(JSlider.VERTICAL, 1, 1000, 500);
```

在该滑块中，用来选择数字的组件的初始位置是在500处。

要在滑块上显示标签，必须设置标签包含的信息。调用滑块的 `setMajorTickSpacing (int)` 和 `setMinorTickSpacing (int)` 方法指定标签上刻度的密度。主刻度使用的线条比次刻度粗。

在设置了刻度的密度之后，可以使用 `true` 作为参数来调用滑块的 `setPaintTicks (boolean)` 方法。还可以使用 `true` 作为参数来调用滑块的 `setPaintLabels (boolean)` 方法，显示每一个主刻度的数值。

16.2 变更监听器

为了监视用户使用滑块进行的输入，必须有实现了 `ChangeListener` 接口的类，该接口位于 `javax.swing.event` 包中。该接口只包含一个方法，格式如下：

```
public void stateChanged(ChangeEvent event); {  
    // statements to handle the event  
}
```

要将对象注册为变更监听器，可调用滑块所属的容器的 `addChangeListener (Object)` 方法。滑块移动时，将调用监听对象的 `stateChanged()` 方法。

该方法将 `ChangeEvent` 对象作为参数，用于指出其值发生了变化的滑块组件。调用该对象的 `getSource()` 方法并将返回的对象转换为 `JSlider`，如下面的语句所示：

```
JSlider changedSlider = (JSlider) event.getSource();
```

在本例中，`event` 是 `ChangeEvent` 对象，用作方法 `stateChanged()` 的参数。

滑块移动时，将发生变更事件。该事件在滑块开始移动时发生，直到用户松开滑块为止。因此，可能要等到滑块停止移动后才调用 `stateChange()` 方法。

要确定滑块是否在移动，可调用其 `getValueIsAdjusting()` 方法。如果滑块在移动，该方法返回 `true`，否则返回 `false`。

接下来的项目将演示这种技巧，这是一个使用3个滑块来选择颜色的Java应用程序。在Java中，颜色是使用 `Color` 类创建的，这个类位于 `java.awt` 包中。

一种创建 `Color` 对象的方法是指定颜色中红、绿和蓝分量的值。每个值都可以是0～255的整数，255表示最大。

例如，下面的语句创建一个代表奶油糖色的Color对象：

```
Color butterscotch = new Color(255, 204, 128);
```

创建该Color对象时使用的红色分量为255，因此包含最大的红色分量。它还包含较大绿色分量和蓝色分量。

程序清单16.1所示的ColorSlider应用程序包含3个滑块、用于标记滑块的3个标签以及一个用于显示颜色的面板。创建一个名为ColorSliders的Java空文件，然后输入程序清单16.1中的全部文本，并保存。

程序清单16.1 ColorsSliders.java程序的完整源代码

```
1: package com.java24hours;
2:
3: import javax.swing.*;
4: import javax.swing.event.*;
5: import java.awt.*;
6:
7: public class ColorSliders extends JFrame implements
ChangeListener {
8:     ColorPanel canvas;
9:     JSlider red;
10:    JSlider green;
11:    JSlider blue;
12:
13:    public ColorSliders() {
14:        super("Color Slide");
15:        setLookAndFeel();
16:        setSize(270, 300);
17:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18:        setVisible(true);
19:
20:        canvas = new ColorPanel();
```

```
21:         red = new JSlider(0, 255, 255);
22:         green = new JSlider(0, 255, 0);
23:         blue = new JSlider(0, 255, 0);
24:
25:         red.setMajorTickSpacing(50);
26:         red.setMinorTickSpacing(10);
27:         red.setPaintTicks(true);
28:         red.setPaintLabels(true);
29:         red.addChangeListener(this);
30:
31:         green.setMajorTickSpacing(50);
32:         green.setMinorTickSpacing(10);
33:         green.setPaintTicks(true);
34:         green.setPaintLabels(true);
35:         green.addChangeListener(this);
36:
37:         blue.setMajorTickSpacing(50);
38:         blue.setMinorTickSpacing(10);
39:         blue.setPaintTicks(true);
40:         blue.setPaintLabels(true);
41:         blue.addChangeListener(this);
42:
43:         JLabel redLabel = new JLabel("Red: ");
44:         JLabel greenLabel = new JLabel("Green: ");
45:         JLabel blueLabel = new JLabel("Blue: ");
46:         GridLayout grid = new GridLayout(4, 1);
47:         FlowLayout right = new FlowLayout(FlowLayout.RIGHT);
48:         setLayout(grid);
49:
50:         JPanel redPanel = new JPanel();
51:         redPanel.setLayout(right);
52:         redPanel.add(redLabel);
53:         redPanel.add(red);
54:         add(redPanel);
55:
56:         JPanel greenPanel = new JPanel();
57:         greenPanel.setLayout(right);
58:         greenPanel.add(greenLabel);
59:         greenPanel.add(green);
60:         add(greenPanel);
61:
62:         JPanel bluePanel = new JPanel();
63:         bluePanel.setLayout(right);
64:         bluePanel.add(blueLabel);
65:         bluePanel.add(blue);
66:         add(bluePanel);
67:         add(canvas);
68:
69:         setVisible(true);
70:     }
71:
```

```

72:     public void stateChanged(ChangeEvent event) {
73:         JSlider source = (JSlider) event.getSource();
74:         if (source.getValueIsAdjusting() != true) {
75:             Color current = new Color(red.getValue(),
76:                                     ➡ green.getValue(),
77:                                     blue.getValue());
78:             canvas.setColor(current);
79:             canvas.repaint();
80:         }
81:     }
82:     private void setLookAndFeel() {
83:         try {
84:             UIManager.setLookAndFeel(
85: "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
86:             );
87:         } catch (Exception exc) {
88:             // ignore error
89:         }
90:     }
91:
92:     public static void main(String[] arguments) {
93:         ColorSliders cs = new ColorSliders();
94:     }
95: }
96:
97: class ColorPanel extends JPanel {
98:     Color background;
99:
100:     ColorPanel() {
101:         background = Color.red;
102:     }
103:
104:     public void paintComponent(Graphics comp) {
105:         Graphics2D comp2D = (Graphics2D) comp;
106:         comp2D.setColor(background);
107:         comp2D.fillRect(0, 0, getSize().width,
108: getSize().height);
109:     }
110:
111:     void changeColor(Color newBackground) {
112:         background = newBackground;
113:     }

```

运行该程序时，结果将会如图16.1所示。其中，一个框架包含3个表示红、绿、蓝分量的滑块以及框架底部的面板。

调整每个滑块的值，以更改显示的颜色。

By the Way

注意

在图16.1中，该应用程序用于创建North Texas中间绿，其红色分量为50，绿色分量为150，蓝色分量为50。这种颜色激励北德克萨斯大学校友在体育活动上要超越自己，并做出可怕的鹰爪手势表示将恐惧留给对手（红色分量255，绿色分量265，蓝色分量0）。

16.3 使用图像图标和工具栏

为了提升图形用户界面的视觉效果，最简单的一种方法是使用图标（小型图像）来标识按钮和界面的其他部分。

Swing类库中有很多组件，可以使用图像（而不是文本）来标记组件，为此可使用javax.swing包中的ImageIcon类。

可以通过调用构造函数ImageIcon(String)，使用计算机中的文件来创建ImageIcon。该方法的参数可以是文件名，也可以是文件的位置和名称，所下面的示例所示：

```
ImageIcon stopSign = new ImageIcon("stopsign.gif");  
ImageIcon saveFile = new ImageIcon("images/savefile.gif");
```

Watch Out!

警告:

尽管有些操作系统使用字符\来分隔文件夹和文件名，但ImageIcon的构造函数要求使用字符/。

用于创建图像图标的图形文件必须为GIF、JPEG或PNG格式。大多数为GIF格式，因为它非常适合用于显示只有几种颜色的小图形。

ImageIcon的构造函数将立即从文件中加载整个图像。

可以通过调用构造函数JLabel(ImageIcon)和JButton(ImageIcon)，将图标用于标签和按钮，如下例所示：

```
ImageIcon siteLogo = new ImageIcon("siteLogo.gif");
JLabel logoLabel = new JLabel(siteLogo);
ImageIcon searchWeb = new ImageIcon("searchGraphic.gif");
JButton search = new JButton(searchWeb);
```

有几种组件可同时包含图标和文本，下面的语句创建一个同时包含文本和图标的按钮：

```
JButton refresh = new JButton("Refresh",
    "images/refreshIcon.gif");
```

图像图标经常用在工具栏中，工具栏是一种将多个组件放在一行或单列中的容器。

工具栏是使用JToolBar类创建的，可被设计为允许用户在图形用户界面中移动它们。这被称为“停靠”，而这种组件也称为“可停靠的工具栏”。

可以调用下面的构造函数之一来创建工具栏。

- JToolBar(): 创建沿水平方向排列组件的工具栏。
- JToolBar(int): 创建沿指定方向排列组件的工具栏，指定的方向可以是SwingConstants.HORIZONTAL或SwingConstants.VERTICAL。

将组件加入到工具栏的方式与加入到其他容器相同：调用add(Component)方法并将要加入的组件作为参数。

对于可停靠的工具栏，必须放在使用BorderLayout布局管理器的容器中。这种布局将容器划分为北、南、东、西、中5个区域。然而，在使用可停靠的工具栏时，容器应只使用其中的两个区域：中央区域和一个方向区域。

工具栏应加入到方向区域中，下面的语句创建一个可停靠的垂直工具栏，其中包含3个图标按钮：

```
JPanel pane = new JPanel();  
BorderLayout border = new BorderLayout();  
pane.setLayout(border);
```



```
JToolBar bar = new JToolBar(SwingConstants.VERTICAL);
ImageIcon play = new ImageIcon("play.gif");
JButton playButton = new JButton(play);
ImageIcon stop = new ImageIcon("stop.gif");
JButton stopButton = new JButton(stop);
ImageIcon pause = new ImageIcon("pause.gif");
JButton pauseButton = new JButton(pause);
bar.add(playButton);
bar.add(stopButton);
bar.add(pauseButton);
add(bar, BorderLayout.WEST);
```

接下来将创建的项目Tool是一个Java应用程序，它包含图像图标和一个可停靠的工具栏。创建一个名为Tool的Java空文件，然后输入程序清单16.2中的全部文本，并保存。

程序清单16.2 Tool.java程序的完整源代码

```
1: package com.java24hours;
2:
3: import java.awt.*;
4: import javax.swing.*;
5:
6: public class Tool extends JFrame {
7:     public Tool() {
8:         super("Tool");
9:         setLookAndFeel();
10:        setSize(370, 200);
11:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12:
13:        // build toolbar buttons
14:        ImageIcon image1 = new ImageIcon("newfile.gif");
15:        JButton button1 = new JButton(image1);
16:        ImageIcon image2 = new ImageIcon("openfile.gif");
17:        JButton button2 = new JButton(image2);
18:        ImageIcon image3 = new ImageIcon("savefile.gif");
19:        JButton button3 = new JButton(image3);
20:
21:        // build toolbar
22:        JToolBar bar = new JToolBar();
23:        bar.add(button1);
```

```

24:         bar.add(button2);
25:         bar.add(button3);
26:
27:         // build text area
28:         JTextArea edit = new JTextArea(8, 40);
29:         JScrollPane scroll = new JScrollPane(edit);
30:
31:         // create frame
32:         BorderLayout border = new BorderLayout();
33:         setLayout(border);
34:         add("North", bar);
35:         add("Center", scroll);
36:         setVisible(true);
37:     }
38:
39:     private void setLookAndFeel() {
40:         try {
41:             UIManager.setLookAndFeel(
42: "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
43:         );
44:         } catch (Exception exc) {
45:             // ignore error
46:         }
47:     }
48:
49:     public static void main(String[] arguments) {
50:         Tool frame = new Tool();
51:     }
52: }

```

应用程序Tool需要3个图形文件用于创建工具栏中的图标：
newfile.gif、openfile.gif和savefile.gif。从本书的配套网站
www.java24hous.com中找到第16章的站点页面，然后下载这3个文件，
并保存到Java24项目文件夹中（或者是你自己在NetBeans中指定的Java
项目文件夹）。

***Did you
Know?***

提示

没有找到NetBeans项目文件夹？在NetBeans中打开一个新的项目：选择File->New Project，在弹出的对话框中，将“分类（category）”设置为Java，将“项目类型（project type）”设置为Java application，然后单击Next按钮。The Project Location文本框应该会包含用来存放这些图标的文件夹的位置。

将这些文件导入项目中的一种方式是使用拖放。

在Project面板中，单击File选项卡。

拖动3个图形文件（一次一个或将这3个作为一组）到Java24文件夹中。

NetBeans将这些文件导入到项目最顶层的文件夹中。

图16.2和图16.3是该程序运行时的两个不同的屏幕截图，工具栏已从初始位置（见图16.2）移到界面的另一边（见图16.3）。



图16.2 使用包含工具栏的应用程序



图16.3 将工具栏停靠到一个新的位置

运行该应用程序，然后尝试移动工具栏。按住工具栏的手柄将其拖曳到文本区域的另一边，可以实现工具栏的移动。松开鼠标后，工具栏将沿文本区域的边缘放置，而文本区域将自动调整位置为工具栏腾出空间。

By the Way

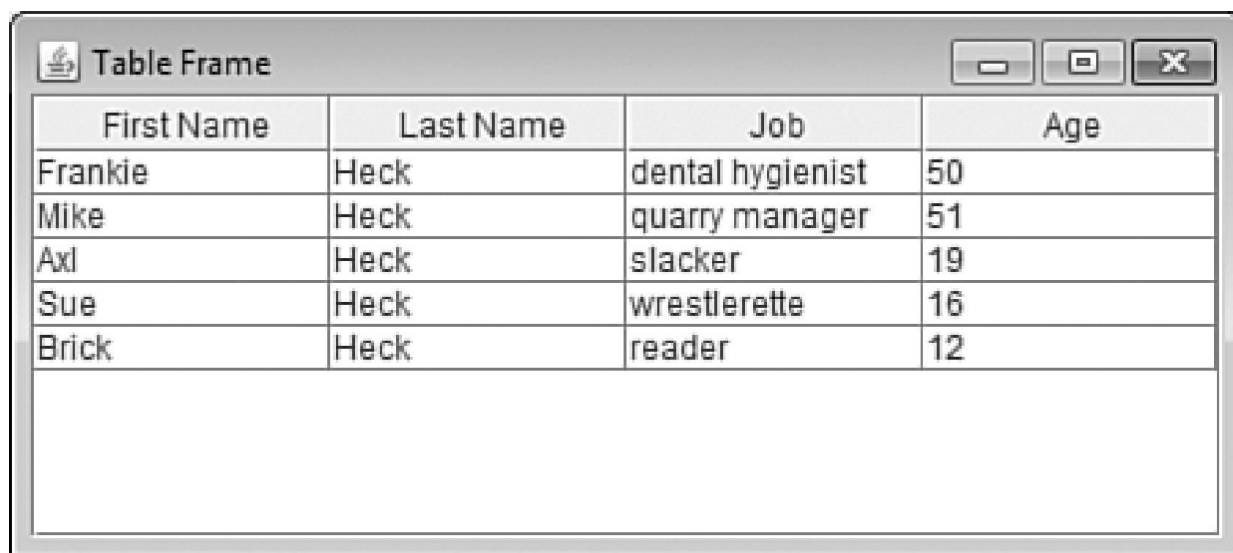
注意

也可以将可停靠的工具栏完全拖放到用户界面外，这将打开一个包含工具栏的新窗口。

16.4 表

本章要介绍的最后一个用户界面组件是最复杂的。JTable类以电子表格的形式（带有行和列）来表示信息。位于表单元格中的信息可以进行编辑，表格的列可以重新排列，而且还可以调整列的宽度。这听起来要做好多工作，但是大部分都已经为你完成了。这些功能内置到属于java.swing包中的一个类中。

JTable组件的一个示例如图16.4所示。这也是本章创建的最后一个项目。



First Name	Last Name	Job	Age
Frankie	Heck	dental hygienist	50
Mike	Heck	quarry manager	51
Axl	Heck	slacker	19
Sue	Heck	wrestlerette	16
Brick	Heck	reader	12

图16.4 包含5行4列的个人信息表

使用表的最简单的方式就是直接将数据存储在表中。

在使用本节介绍的技术时，第一步是创建一个字符串数组，每一列都带有名字，如下面的代码所示：

```
String[] columnLabels = { "First Name", "Last Name",  
    "Job", "Age" };
```

表中的单元格可以存放不同的数据类型或事件对象。但是每一列存放的类型则必须相同。在即将创建的项目中，每一行有3个字符串（名字、姓氏和职位）和一个表示年龄的整数。因此，每一行的单元格从左到右必须依次存放3个字符串和一个整数。

填充表格的下一步是创建一个二维对象数组，它包含要存储的数据。用来完成该目的的代码类似于第9章中用到的：声明一个数组，然后给它们赋予初始值。**columnLabels**数组就是这样创建的。

下面的代码在**Object [][]**数组中创建了表格中使用的数据：

```
Object[][] tableData = {  
    {  
        "Frankie", "Heck", "dental hygienist", 50  
    },  
    {  
        "Mike", "Heck", "quarry manager", 51  
    },  
    {  
        "Axl", "Heck", "slacker", 19  
    },  
    {  
        "Sue", "Heck", "wrestlerette", 16  
    },  
    {  
        "Brick", "Heck", "reader", 12  
    }  
};
```

上面的代码是以多行形式显示的一行语句，它创建了一个被组织为5行4列的二维对象数组。每一行定义了某个人的信息：存放名字、姓氏和职位的字符串，随后是一个表示年龄的整数。

By the Way

注意

读者可能想知道，像50或16这样的数字怎么能用来创建对象数组中的一个条目？该代码使用了Java的封装和拆封功能，自动在对象和原始数据类型之间进行转换。这里的整数字面值被封装为表示该整数值的对象。

在这段代码中有大量的空格和标点符号，所以要注意花括号“{”、“}”所在的位置，以及逗号的位置。这里的空格不重要，但是标点符号必须与这里的完全一致。

数组的每一行都被分配了数据，这些数据放在花括号之间，而且使用逗号进行分割。下面是格式略有不同的一行数据。

```
{ "Sue", "Heck", "wrestlerette", 16 }
```

为了让读者进一步了解该定义，我们将其放到一行代码中，创建一个一维的对象数组：

```
Object[] row1 = { "Sue", "Heck", "wrestlerette", 16 };
```

对于二维数组来说，定义了每一行的语句块是使用逗号分割的。所以，有一个语句块来定义**Frankie**，然后是一个逗号；然后是定义**Mike**的语句块，然后是一个逗号；然后是定义**Axl**的语句块，然后是一个逗号；然后是定义**Sue**的语句块，然后是一个逗号；最后是定义**Brick**的语句块。在**Java**代码中任何以逗号分割的列表，在最后一个条目后不再需要逗号。

在有了一个表示列名的字符串宿主和一个表示表中数据的对象数组后，可以使用表中数据和列名作为构造函数的参数来创建**JTable**对象：

```
JTable table = new JTable(tableData, columnLabels);
```

表中包含的信息通常无法在一个图形用户界面中全部显示出来，因此需要将表放到一个滚动面板中，以支持滚动查看。

通过调用表格的**setFillsViewportHeight(Boolean)**方法，并使用**true**作为参数，可以让表格占用图形用户界面中所有可用的高度：

```
table.setFillsViewportHeight(true);
```


下面要创建的TableFrame应用程序将上述所有内容整合到了一起。在NetBeans中创建一个新的Java空文件，然后输入程序清单16.3中的多有内容。

程序清单16.3 TableFrame.java的完整源代码

```
1: package com.java24hours;
2:
3: import javax.swing.JFrame;
4: import javax.swing.JScrollPane;
5: import javax.swing.JTable;
6:
7: public class TableFrame extends JFrame {
8:     public TableFrame() {
9:         super("Table Frame");
10:        String[] columnLabels = { "First Name", "Last Name",
11:                                   "Job", "Age" };
12:        Object[][] tableData = {
13:            { // row 1
14:                "Frankie", "Heck", "dental hygienist", 50
15:            },
16:            { // row 2
17:                "Mike", "Heck", "quarry manager", 51
18:            },
19:            { // row 3
20:                "Axl", "Heck", "slacker", 19
21:            },
22:            { // row 4
23:                "Sue", "Heck", "wrestlerette", 16
24:            },
25:            { // row 5
26:                "Brick", "Heck", "reader", 12
27:            }
28:        };
29:        JTable table = new JTable(tableData, columnLabels);
30:        JScrollPane scrollPane = new JScrollPane(table);
31:        table.setFillsViewportHeight(true);
32:        add(scrollPane);
33:        setSize(450, 200);
34:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35:        setVisible(true);
36:    }
37:
38:    public static void main(String[] arguments) {
39:        TableFrame frame = new TableFrame();
```

```
40:     }  
41: }
```

运行该应用程序，将看到如图16.4所示的界面。可以执行下面的3件事情来测试表格中内置的功能。

双击表中的一个单元格。在该单元格中出现的光标可以让你输入新的值来替代现有的值。

单击一个列标签，向左或向右拖动，然后松开。列的顺序重新调整，但是单元格中的信息不变。

将鼠标光标悬停在分割两个列标签的线上，光标变成一个向右的箭头。将这条分割线移动到一个新的位置，从而调整两个列的宽度。

By the Way

注意

使用数组的方式将数据放到JTable组件中的一个缺点是，它会将每一个单元格都视为字符串，而且允许用户修改每一个单元格，以及随意替换其值。要知道，TableFrame应用程序中有一列使用的是数字，而这会将其替换为非整数和非数字的值。

要对能输入到表格中的数值进行更多控制，并且完全阻止对某些列的访问，可以使用javax.swing.table包中的TableModel接口。表模型是一个提供与表格内容及其修改方式相关的信息的对象。

16.5 总结

本章是介绍Swing的四章中的最后一章，Swing是Java语言中支持GUI软件的部分。

尽管Swing目前是Java类库的最大组成部分，但其中大部分类的用法相似。在知道如何创建组件、如何将组件添加到容器、如何设置容器的布局管理器及如何响应用户输入后，探索该语言时就能够使用众多新的Swing类。

16.6 问与答

问：如何找到Java类库中的其他Swing类？

答：在Oracle的Java官方网站中，有完整的Java类库文档，其网址为<http://download.oracle.com/javase/8/docs/api>。在这里可以查看前面4章介绍的javax.swing、java.swt和java.awt.event包中的类。所有Swing类和接口都有文档，包括它们的构造函数、类变量和实例变量。

16.7 测验

一分耕耘一分收获：回答下面有关滚动面板、图像图标和其他Swing特性的问题，以锻炼你的大脑。

16.7.1 问题

1. ImageIcon类支持哪些图形文件格式？

- a. GIF。
 - b. GIF和JPEG。
 - c. GIF、PNG和JPEG。
2. JSlider对象的getValueIsAdjusting()方法有什么功能？
- a. 判断滑块的值是否已改变？
 - b. 判断滑块的值是否正在改变？
 - c. 不做什么，提供该方法主要是对超类不满意。

16.7.2 答案

- 1. c. 从Java 1.3开始，ImageIcon开始支持PNG格式。
- 2. b. 如果滑块正在移动，getValueIsAdjusting()方法将返回true，否则返回false。

16.8 练习

为测试是否掌握了Swing，请完成下面的练习：

- 在前面3章创建的Java应用程序中，添加一个工具栏；
- 创建一个表格，用来显示某一支股票至少5天的最高值、最低值、交易量和相应日期。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站
www.java24hours.com。

第17章 在数据结构中存储对象

本章介绍如下内容：

- 创建一个数据列表；
- 在列表中增删条目；
- 使用泛型提升列表的可靠性；
- 在列表中搜索对象；
- 遍历列表的内容；
- 创建键值之间的哈希映射；
- 在映射中增删条目；
- 检索映射条目的键；
- 检索映射条目的值；
- 遍历映射的键和值。

程序员都是储物狂。

在计算机编程中，你花费的大量的时间来收集信息，然后寻找地方将其存储起来。你收集来的数据都有其原始的数据类型，比如float或某一个特定类的对象。数据可以从磁盘上读取而来，可以从Internet服务器上获取，可以由用户输入，还可以通过其他方式来获取。

在拥有了数据之后，在程序在Java虚拟机中运行时，你必须决定要将数据放在何处。在数据类型或类上相关的多个条目可以存储在一个数组中。

对许多用途来讲，这已经足够了，但是随着你的程序日渐成熟，你的存储需求也会随之增长。

在本章，你将学到Java中用来存放信息的一些类：数组列表和哈希映射。

17.1 数组列表

你在第9章学习了数组，这是一种在程序中处理成变量组或对象组的便捷方式。数组对Java而言非常重要，而且与整数和字符一样，它也是一种内置的数据类型。数组将具有相同数据类型或类的元素打包在一起。

数组非常有用，但是它面临的一个事实是，其大小不能调整。如果在创建一个数组时，规定了存储90个元素，它的大小就固定了下来，不能再增大或减小（对此有一个花哨的说法——数组是“一成不变（immutable）”的）。

在java.util包中有一个类可以实现数组所有的功能，而且没有数组的大小限制，它就是ArrayList。

数组列表是一个存储同一类对象或具有共同超类的对象的数据结构。在程序运行时，列表可以根据需要调整大小。

创建数组列表最简单的方法是调用其不带参数的构造函数：

```
ArrayList servants = new ArrayList();
```

在创建数组列表时，可以指定一个初始的容量（大小），这为列表能存放多少个元素提供了指导。该容量作为一个整型参数传递给构造函数：

```
ArrayList servants = new ArrayList(30);
```

尽管这看起来像是创建了一个具有确切大小的数组，但是容量只是一个提示。如果超出了这个值，数组列表会进行相应调整，而且继续正常运行（你估计的容量越准确，程序的效率就越高）。

列表用来存放属于同一类或共享同一个超类的对象。

By the Way

注意

如果你使用过之前版本的Java，则可能听说过vector（矢量），而且好奇为vector为什么与数组列表这么像。vector是功能与ArrayList类几乎相同的另外一种数据结构，而且位于java.util.Vector包中。两者之间最重主要的区别是vector需要同步，因此任何使用了vector的类，运行起来都很慢。出于这个原因，数组列表成为了更好的选择。

当创建数据列表时，你要知道列表打算存储的类或超类。这可以在构造函数中的“<”和“>”符号内来指定。而“<”和“>”符号是一种称为

泛型的语言特性。下面是一个改进后的列表构造函数，用来存放String对象：

```
ArrayList<String> servants = new ArrayList<String>();
```

要添加对象，可将该对象作为参数调用数组列表的add(Object)方法。下面是添加5个字符串的语句。

```
servants.add("Bates");  
servants.add("Anna");  
servants.add("Thomas");  
servants.add("Mrs. O'Brien");  
servants.add("Daisy");
```

每一个元素都被添加到列表的末尾，因此servants中的第一个字符串是“Bates”，而最后一个则是“Daisy”。

这里还有一个相应的remove(Object)方法，用来将对象从列表中移除。

```
servants.remove("Mrs. O'Brien");
```

数组列表的大小是它当前存储的元素的个数。可以调用列表的 `size()` 方法来获取这一信息，该方法返回一个整数：

```
int servantCount = servants.size();
```

当使用了泛型来指定列表包含的类时，使用 `for` 循环来迭代列表中的每一个元素就很简单了：

```
for (String servant : servants) {  
    System.out.println(servant);  
}
```

`for` 的第一个参数是一个变量，它应该存储一个元素。第二个参数是数组列表。其他数据结构也可使用相同的循环。

`add(Object)` 方法将对象存储在列表的末尾。也可以在列表中指定对象要存放的位置，从而将对象存放到列表中。这会用到 `add(int, Object)` 方法，该方法的第一个参数表示要存放的位置。

```
ArrayList<String> aristocrats = new ArrayList<String>();  
aristocrats.add(0, "Lord Robert");  
aristocrats.add(1, "Lady Mary");  
aristocrats.add(2, "Lady Edith");  
aristocrats.add(3, "Lady Sybil");  
aristocrats.add(0, "Lady Grantham");
```

上面这个例子中，最后一条语句将“Lady Grantham”放在的列表的首位（即放在了“Lord Robert”和其他对象的上面），而非最后。

作为第一个参数指定的位置必须不能大于列表的size()方法返回的值。如果将“Lord Robert”添加到位置1而非位置0，程序会因为IndexOutOfBoundsException而出错。

通过将对象的位置指定为remove(int)的参数，也可以将其从列表中移除：

```
aristocrats.remove(4);
```

通过调用get(int)方法以及元素在列表中出现的位置，可以检索该元素。下面这个for循环将每一个字符串提取出来并显示：

```
for (int i = 0; i < aristocrats.size(); i++) {  
    String aristocrat = aristocrats.get(i);  
    System.out.println(aristocrat);  
}
```

通常，我们有必要确定一个数组列表中是否包含一个特定对象。这可以通过调用列表的indexOf(Object)方法以及将该对象作为参数来实现。该方法将返回对象的位置，如果没有找到，则返回-1。

```
int hasCarson = servants.indexOf("Carson");
```

本章的第一个项目会在一个简单的游戏中使用这些技术。在这个游戏中，我们在10×10的网格中朝着(x,y)点开枪。有些点包含了目标，而有些则没有包含。

目标由java.awt包中的Point类来表示。通过调用Point(int, int)构造函数，同时将x和y坐标作为两个参数，可以创建一个点。

下面这条语句在（5,9）处创建了一个点：

```
Point p1 = new Point(5,9);
```

这里是一个10×10的网格，其中一个点标标记为X，空区域标记为一个点号（.）。

Output ▼

	1	2	3	4	5	6	7	8	9
1
2	X
3
4
5
6
7
8
9

列从左到右表示x轴，行从上到下表示y轴。

在该项目开始之前，你要知道数组列表是如何存放字符串的。数组列表可以存放**Point**或其他类型的对象。下面的语句创建了一系列点：

```
ArrayList<Point> targets = new ArrayList<Point>();
```

Java编译器不允许**Point**或其子类之外的任何其他类添加到数组列表中。

在NetBeans或其他编程工具中，创建名为**Battlepoint**的Java文件，然后将**com.java24hours**作为其包。输入程序清单17.1中的所有源代码。

程序清单17.1 Battlepoint.java的完整版本

```
1: package com.java24hours;  
2:  
3: import java.awt.*;  
4: import java.util.*;
```

```

5:
6: public class Battlepoint {
7:     ArrayList<Point> targets = new ArrayList<Point>();
8:
9:     public Battlepoint() {
10:         // create targets to shoot at
11:         createTargets();
12:         // display the game map
13:         showMap();
14:         // shoot at three points
15:         shoot(7,4);
16:         shoot(3,3);
17:         shoot(9,2);
18:         // display the map again
19:         showMap();
20:     }
21:
22:     private void showMap() {
23:         System.out.println("\n 1 2 3 4 5 6 7 8 9");
24:         for (int column = 1; column < 10; column++) {
25:             for (int row = 1; row < 10; row++) {
26:                 if (row == 1) {
27:                     System.out.print(column + " ");
28:                 }
29:                 System.out.print(" ");
30:                 Point cell = new Point(row, column);
31:                 if (targets.indexOf(cell) > -1) {
32:                     // a target is at this position
33:                     System.out.print("X");
34:                 } else {
35:                     // no target is here
36:                     System.out.print(".");
37:                 }
38:                 System.out.print(" ");
39:             }
40:             System.out.println();
41:         }
42:         System.out.println();
43:     }
44:
45:     private void createTargets() {
46:         Point p1 = new Point(5,9);
47:         targets.add(p1);
48:         Point p2 = new Point(4,5);
49:         targets.add(p2);
50:         Point p3 = new Point(9,2);
51:         targets.add(p3);
52:     }
53:
54:     private void shoot(int x, int y) {
55:         Point shot = new Point(x,y);

```

```
56:         System.out.print("Firing at (" + x + ", " + y + ") ...  
");  
57:         if (targets.indexOf(shot) > -1) {  
58:             System.out.println("you sank my battlepoint!");  
59:             // delete the destroyed target  
60:             targets.remove(shot);  
61:         } else {  
62:             System.out.println("miss.");  
63:         }  
64:     }  
65:  
66:     public static void main(String[] arguments) {  
67:         new Battlepoint();  
68:     }  
69: }
```

应用程序Battlepoint中的注释描述了构造函数和每一部分以及程序中重要的条件逻辑部分。

应用程序将目标创建为3个Point对象，然后将它们添加到一个数组中（第45～52行代码）。显示一个标记了这些目标的地图（第22～43行代码）。

接下来，调用shoot(int, int)方法向3个目标开火（第54～64行代码）。应用程序会报告是否击中其中一个目标。如果击中了，则将该目标从数组列表中删除。

最后，再次显示地图，同时应用程序终止运行。

该程序的输出入图17.1所示。

```
Output - Java21 (run) x Tasks
run:

  1  2  3  4  5  6  7  8  9
1  .  .  .  .  .  .  .  .  .
2  .  .  .  .  .  .  .  .  X
3  .  .  .  .  .  .  .  .  .
4  .  .  .  .  .  .  .  .  .
5  .  .  .  X  .  .  .  .  .
6  .  .  .  .  .  .  .  .  .
7  .  .  .  .  .  .  .  .  .
8  .  .  .  .  .  .  .  .  .
9  .  .  .  .  X  .  .  .  .

Firing at (7,4) ... miss.
Firing at (3,3) ... miss.
Firing at (9,2) ... you sank my battlepoint!

  1  2  3  4  5  6  7  8  9
1  .  .  .  .  .  .  .  .  .
2  .  .  .  .  .  .  .  .  .
3  .  .  .  .  .  .  .  .  .
4  .  .  .  .  .  .  .  .  .
5  .  .  .  X  .  .  .  .  .
6  .  .  .  .  .  .  .  .  .
7  .  .  .  .  .  .  .  .  .
8  .  .  .  .  .  .  .  .  .
9  .  .  .  .  X  .  .  .  .

BUILD SUCCESSFUL (total time: 0 seconds)
```

图17.1 将Point对象放到一个数组列表汇总，然后向其开火

在图17.1顶部的地图上可以看到有3个目标。在其中一个被击中后，从地图上移除了。图17.1底部的地图反映了这一变化。

当一个目标被击中后，通过调用`remove(Object)`方法，并使用射击点作为参数，将目标从`targets`数据列表中移除。

Watch Out!

警告:

在Battlepoint应用程序的shoot()方法中，将要从数组列表中移除的Point对象表示被击中的那个目标。它的(x,y)坐标与被击中的目标相同。

17.2 哈希映射

在编程中，使用一条信息来访问另一条信息的事情很常见。在数据结构中，数组列表是实现该功能的一个最简单的例子，它使用索引号从列表中检索一个对象。下面是从aristocrate数组列表中拉取第一个字符串的例子：

```
String first = aristocrats.get(0);
```

数组还可以使用索引号检索数组中的每一个项目。

哈希映射是Java中的一种数据结构，它使用对象来检索另外一个对象。第一个对象是键，第二个对象是值。它们是作为java.util包中的HashMap类来实现的。

哈希映射指的是如何将键映射为值。这种数据结构的一个例子是电话簿。一个人的名字（字符串）可以用来检索他的电话号码。

By the Way

注意

正如数组列表有另外一个称为**Vector**的类，该类以资源密集型的方式实现与数组列表同样的功能一样，哈希映射有一个**Hashtable**。哈希表需要进行同步，而哈希映射则不需要。使用了哈希表的类，其运行时间更长，因此哈希映射称为更优的选择。

可以通过调用其无参数的构造函数来创建哈希映射：

```
HashMap phonebook = new HashMap();
```

在一个新的哈希映射中，可以指定两件事情：初始容量和负载因子。这两件事情用来控制哈希映射的效率。它们是使用两个参数来设置的，如下所示：

```
HashMap phonebook = new HashMap(30, 0.7F);
```

容量是可以存储哈希映射值的存储桶的个数。负载因子是在自动增加容量之前，可以使用的存储桶的个数，该值是一个浮点数，其值为0（空）～1.0（满）。因此，0.7意味着当存储桶的使用量为70%时，

将自动增加容量。默认的容量值为16，负载因子为0.75，通常来讲，这已经足够的。

应该使用泛型来指明键和值的类。它们放在“<”和“>”字符内，而且类名使用逗号分隔，如下所示：

```
HashMap<String, Long> phonebook = new HashMap<>();
```

这创建了一个名为phonebook的哈希映射，而且其键为字符串，其值为Long对象。第二组“<”和“>”字符内为空，它假定这里的类名与语句中第一组“<”和“>”内的类名相同。

通过调用带有两个参数（键和值）的put(Object, Object)方法，将对象存储到哈希映射中。

```
phonebook.put("Butterball Turkey Line", 8002888372);
```

这将一个键为“Butterball Turkey Line”，值8002888374为Long对象的条目存储到哈希映射中。其中，值8002888374为Butterball Turkey Line服务的电话号码。

通过调用get(Object)方法，同时将键作为其唯一的参数，可以从映射中检索对象。

```
int number = phonebook.get("Butterball Turkey Line");
```

如果没有发现匹配该键的值，`get()`方法将返回`null`。在前面的列子中，这将引发一个问题，因此`null`不是一个合适的`long`值。

By the Way

注意

这些语句使用`long`值将`Long`对象放到哈希映射中。在Java早期的版本中，这将引发一个错误。因此`long`这样的原始数据类型不能用于需求是对象的情况中。

由于自动封装和接封装的出现，现在它不再是一个错误。自动封装和接封装是Java的一个特性，它能够自动在原始类型和等价的对象类之间进行转换。当Java编译器看到8002888372这样的`long`数值时，它会将其转换为相应的`Long`对象。

处理这一潜在问题的另外一种方式是调用`getOrDefault(Object, Object)`。如果作为第一个参数的键没有被找到，则默认范围第二个参数，如下面的语句所示：

```
int number = phonebook.getOrDefault("Betty Crocker", -1);
```

如果在映射中找到了匹配“Betty Crocker”的一个数值，则返回该数值；否则，返回-1。

有两个方法可以用来指示一个键或值是否在映射中存在：`containsKey(Object)`和`containsValue(Object)`。它们将返回一个为true或false的布尔值。

与数组列表一样，哈希映射也有一个`size()`方法来指示数据结构中条目的个数。

通过使用一个条目集合（`entry set`），可以对一个映射执行遍历。条目集合是映射中所有条目的集合。`entryset()`将这些条目作为一个Set对象返回（使用java.util中的Set接口）。

集合中的每一个条目使用`Map.Entry`来表示，这是java.util包Map类中的一个内部类。当有了一个Entry对象时，可以调用其`getKey()`方法来检索键，调用`getValue()`方法来检索值。

下面的for循环语句使用条目集合和条目来访问phonebook哈希映射中的所有键和所有值：

```
for (Map.Entry<String, Font> entry : map.entrySet()) {  
    String key = entry.getKey();  
    Font value = entry.getValue();  
    // ...  
}
```

FontMapper项目将所有这些内容整合到一起，它使用一个哈希映射来管理一组字体。

java.awt包中的Font类用来创建字体，并使用它们在图形用户界面中显示文本。字体包含字体的名称、字体大小，以及是否字体是普通字体、加粗字体还是斜体字体。

哈希映射可以包含任何对象。打开NetBeans，在com.java24hours包中创建一个Java文件将其命名为FontMapper，然后输入程序清单17.2中的所有代码并保存。

程序清单17.2 FontMapper.java的完整文本

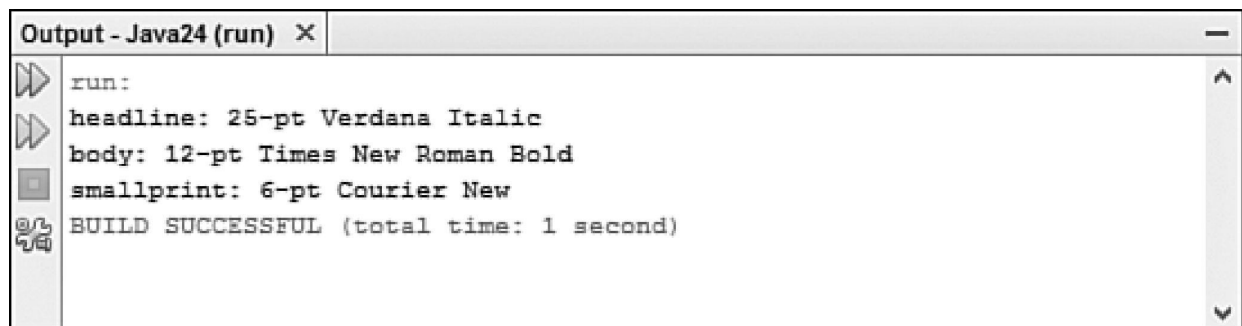
```
1: package com.java24hours;
2:
3: import java.awt.*;
4: import java.util.*;
5:
6: public class FontMapper {
7:     public FontMapper() {
8:         Font courier = new Font("Courier New", Font.PLAIN, 6);
9:         Font times = new Font("Times New Roman", Font.BOLD,
10: );
11:         Font verdana = new Font("Verdana", Font.ITALIC, 25);
12:         HashMap<String, Font> fonts = new HashMap<>();
13:         fonts.put("smallprint", courier);
14:         fonts.put("body", times);
15:         fonts.put("headline", verdana);
16:         for (Map.Entry<String, Font> entry : fonts.entrySet())
17:         {
18:             String key = entry.getKey();
19:             Font value = entry.getValue();
20:             System.out.println(key + ": " + value.getSize() +
21: "-pt "
22:             + value.getFontName());
23:         }
24:     }
25:
26:     public static void main(String[] arguments) {
27:         new FontMapper();
28:     }
29: }
```

```
25:     }  
26: }
```

FontMapper应用程序在第8~10行创建了3个Font对象，然后在第12~14行将添加到fonts哈希映射中。它们在存储到映射中时，还分别带有一个描述字体用途的字符串值：“smallprint”、“body”、“headline”。

第15~20行的for循环语句使用一个条目集合以及集合中的每一个单独的条目来遍历哈希映射。

该应用程序的输出结果如图17.2所示。



```
Output - Java24 (run) X  
run:  
headline: 25-pt Verdana Italic  
body: 12-pt Times New Roman Bold  
smallprint: 6-pt Courier New  
BUILD SUCCESSFUL (total time: 1 second)
```

图17.2 在哈希映射中存储Font对象

17.3 总结

数组列表和哈希映射是java.util包中两种比较有用的数据结构。数组列表类扩展了数组的功能，克服了数组具有固定长度的局限性。哈希映射则可以使用任何类型的对象作为键来检索一个值。

还有位设置（**BitSet**类），它存储的位值为0或1；还有栈（**Stack**），它是一个后进先出的数据集合，与数组列表类似；还有属性（**Properties**），这是一个专门的哈希映射，可以在文件中或其他永久存储的地方存放程序的配置属性。

17.4 问与答

问：“类进行同步”的意思是什么？

答：同步是Java虚拟机确保一个对象的实例变量和方法能够以一致且准确的方式被对象的其他用户来访问的一种机制。

当你学了第19章中的线程之后，就能理解这个概念了。Java程序可以被设计为同时运行多个任务。每一个任务放在自己的线程中运行。

当多个线程访问同一个对象时，对象在每一个线程中的行为必须一致，这至关重要。因此当一个类的方法需要同步时（比如**vector**和哈希表），Java虚拟机必须加紧处理，而且可能会遇到让线程停止运行的错误。

这就是为什么数组列表和哈希映射具有**vector**和哈希表的功能，但是其数据结构却不需要同步的原因，这样运行的效率更高。

17.5 测验

下面的问题可以测试你囤积了多少数据结构相关的知识。

17.5.1 问题

1. 下面哪一个在创建之后不能调整大小?
 - a. 数组列表。
 - b. 数组。
 - c. 哈希映射。

2. 下面哪一个方法可以计算出一个数组列表或哈希映射中条目的数量?
 - a. `length()`。
 - b. `size()`。
 - c. `count()`。

3. 哪种数据类型或类可以在哈希映射中用作键?
 - a. `int`或`Integer`。
 - b. `String`。
 - c. 任何类。

17.5.2 答案

1. b. 数组的大小在创建之时就已经确定，而且无法修改。数组列表和哈希映射都可以根据需要调整大小。

2. b. `size()`方法可以指示数组列表或哈希映射的数据结构中条目的当前个数。

3. a、b或c. 哈希映射可以使用任何类作为键和值。

17.6 练习

你可以在下述练习中应用业已掌握的结构化编程知识：

编写一个程序，它使用了本章介绍的其中一个数据结构来存放公司邮件地址列表，其中每个邮件地址与员工的名字相关联；

扩展FontMapper程序，使其通过命令行参数的形式接收新的字体名称、字号和样式，并在显示键值之前将其添加到哈希映射中。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第18章 处理程序中的错误

本章介绍如下内容：

- 错误为什么称为异常；
- 如何响应Java程序中的异常；
- 创建忽略异常的方法；
- 使用引起异常的方法；
- 创建异常。

导致程序无法正确运行的错误、bug、疏忽和输入错误是软件开发过程的正常组成部分。“正常”可能是用于描述错误的最善意的用词。我在编程时，当不能找出导致程序不能正确运行的难懂错误时，我会用让副总统都脸红的词。

有些错误将被编译器标记出来，进而阻止你创建类；导致程序无法成功运行的其他错误将被解释器发现。Java将错误划分为如下两类。

- 异常（Exception）：表明程序运行时发生了异常情况的事件。
- 错误（Error）：表明解释器遇到问题，但可能与程序无关的事件。

Java程序不能从错误中恢复过来继续运行，所以它不是本章关注的重点。在进行Java编程时，读者可能遇到过OutOfMemoryError的错误。当这类错误发生后，Java程序对其无能为力，只能退出运行。

而异常可以采用某种方式来处理，而且程序也会继续运行。

18.1 异常

虽然刚开始学习它，但通过本书前17章，读者可能已经对异常非常熟悉了。编写的Java程序编译成功但运行时遇到问题时，出现的错误就是异常。

例如，一种常见的编程错误是引用不存在的数组元素，如下面的语句所示：

```
String[] greek = { "Alpha", "Beta", "Gamma" };  
System.out.println(greek[3]);
```

在这个例子中，String数组greek有3个元素。数组的第一个元素编号为0而不是1，因此第一个元素是greek[0]，第二个元素是greek[1]，第三个元素是greek[2]，所以试图显示greek[3]的语句是错误的，因为该元素不存在。上述语句可以成功编译，但运行程序时，Java虚拟机将显示下面的消息并停止运行：

```
Output ▼  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:  
3  
at SampleProgram.main(SampleProgram.java:4)
```

这条消息表明应用程序引发了异常，JVM通过显示错误消息并停止运行程序来指出这一点。

这条错误消息引用了java.lang包中的ArrayIndexOutOfBoundsException类，这是一个异常，该对象用于指出Java程序中发生了异常情况。

Java类遇到异常后，它向类的用户指出错误类型。在这个例子中，类的用户是JVM。

By the Way

注意

有两个术语可用于描述该过程：抛出和捕获。对象抛出异常，以指出发生了异常。这些异常可被其他对象或Java虚拟机捕获。

所有异常都是Exception的子类，Exception位于java.lang包中。正如读者预期的，ArrayIndexOutOfBoundsException类指出访问了数组边界外的元素。

Java中有数百种异常，其中很多表明问题可以通过修改程序得到解决，如数组异常，这些异常类似于编译错误，纠正后就不用担心它会再出现。

其他异常必须使用5个新关键词在程序运行时进行处理：try、catch、finally、throw和throws。

18.1.1 在try-catch块中捕获异常

到目前为止，我们是通过纠正导致异常的问题来处理异常。有时无法以这样的方式来处理异常，因此必须在Java类中处理异常。

为了说明为何这很有用，在新的Java空文件中输入程序清单18.1中的Java应用程序，将其命名为Calculator，然后保存。

程序清单18.1 Calculator.java的源代码

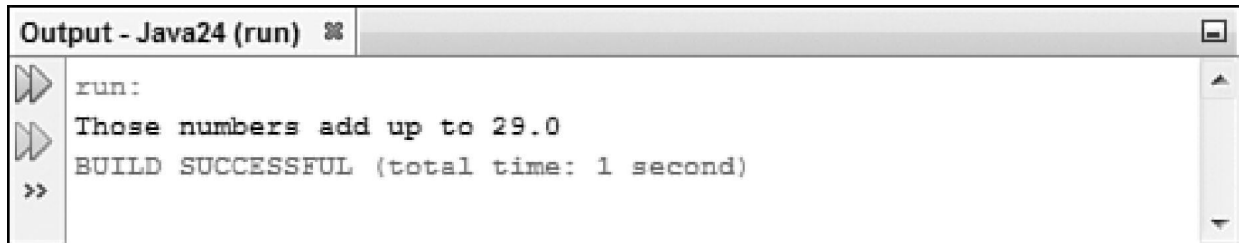
```
1: package com.java24hours;
2:
3: public class Calculator {
4:     public static void main(String[] arguments) {
5:         float sum = 0;
6:         for (String argument : arguments) {
7:             sum = sum + Float.parseFloat(argument);
8:         }
9:         System.out.println("Those numbers add up to " + sum);
10:    }
11: }
```

Calculator应用程序通过命令行参数接受一个或多个数字，然后将它们相加并显示结果。

在Java应用程序中，所有命令行参数都用字符串表示，将它们相加前，程序必须将其转换为浮点数。第7行的Float.parseFloat()方法完成这项任务，并将转换得到的数加到变量sum中。

在运行该应用程序之前，先对命令行参数进行设置：在NetBeans中选择Run->Set Project Configuration->Customize命令，然后手输入7 4 8 1

4 1 4。然后选择Run->Run Main Project来运行该程序，其输出如图18.1所示。



```
Output - Java24 (run) %
run:
Those numbers add up to 29.0
BUILD SUCCESSFUL (total time: 1 second)
```

图18.1 Calculator应用程序的输出

使用不同的数字作为参数运行该应用程序多次，程序都将成功处理，这让读者产生疑惑，这与处理异常有什么关系？

要看到异常，将Calculator应用程序的命令行参数修改为1 3 5x。

这里的第三个参数包含输入错误：数字5后不应是字符x。

Calculator应用程序不知道这是错误，所以将5x同其他数字相加，导致下面的异常：

```
Output ▼
Exception in thread "main" java.lang.NumberFormatException: For
input
string: "5x" at sun.misc.FloatingDecimal.readJavaFormatString
(FloatingDecimal.java:1241)
    at java.lang.Float.parseFloat(Float.java:452)
    at Calculator.main(Calculator.java:7)
Java Result: 1
```

尽管该消息对程序员来说很有用，但却不希望用户看到。如果能够隐藏错误消息，并且在程序中处理这个问题，那是再好不过。

Java程序可以使用try-catch块语句来处理异常，其格式如下：

```
try {  
    // statements that might cause the exception  
} catch (Exception e) {  
    // what to do when the exception occurs  
}
```

对于希望类方法来处理的任何异常，都必须使用一个try-catch块。在catch语句中的Exception对象应是下面3个中的一个：

- 可能发生的异常类；
- 多个异常类，期间使用管道字符“|”隔开；
- 可能发生的多种异常的超类。

try-catch块的try部分应包含可能抛出异常的语句。在应用程序Calculator中，第7行调用了Float.parseFloat(String)方法，当该方法中的参数是不能转换为浮点数的字符串时，将抛出NumberFormatException。

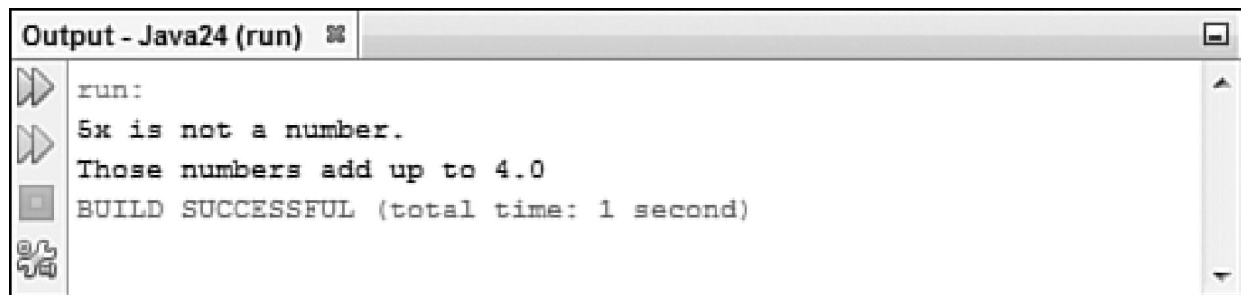
为了改善应用程序Calculators，使其遇到这种错误时不停止运行，可使用一个try-catch块。

创建一个名为NewCalculator的Java空文件，然后输入程序清单18.2中的所有文本。

程序清单18.2 NewCalculator.java程序


```
1: package com.java24hours;
2:
3: public class NewCalculator {
4:     public static void main(String[] arguments) {
5:         float sum = 0;
6:         for (String argument : arguments) {
7:             try {
8:                 sum = sum + Float.parseFloat(argument);
9:             } catch (NumberFormatException e) {
10:                 System.out.println(argument + " is not a
number.");
11:             }
12:         }
13:         System.out.println("Those numbers add up to " + sum);
14:     }
15: }
```

保存该应用程序之后，对程序配置进行自定义，然后使用命令行参数1 3 5x运行该程序，将会看到如图18.2所示的输出。



```
Output - Java24 (run)
run:
5x is not a number.
Those numbers add up to 4.0
BUILD SUCCESSFUL (total time: 1 second)
```

图18.2 NewCalculator应用程序的输出

第7~11行的try-catch块处理Float.parseFloat()方法抛出的NumberFormatException错误。这些异常是在NewCalculator类中捕获的，如果某个参数不是数字，该类将显示一条错误消息。由于在这个类中处理了异常，因此Java解释器不会显示错误消息。可以使用try-catch块来处理与用户输入和其他意外数据相关的问题。

18.1.2 捕获多种不同的异常

`try-catch`块可用来处理几种不同类型的异常，即使这些异常是由不同的语句抛出的。

处理多种异常类的一种方法是在每一个异常中使用一个`catch`语句块，如下面的代码所示：

```
String textValue = "35";
int value;
try {
    value = Integer.parseInt(textValue);
} catch (NumberFormatException exc) {
    // code to handle exception
} catch (ArithmeticException exc) {
    // code to handle exception
}
```

可以在同一个`catch`块中处理多个异常，方法如下：多个异常之间使用管道字符（`|`）隔开，并在最后使用异常变量来结尾。其示例如下。

```
try {
    value = Integer.parseInt(textValue);
} catch (NumberFormatException | ArithmeticException exc) {
    // code to handle exceptions
}
```

如果该代码捕获了`NumberFormatException`或`ArithmeticException`异常，则将其赋值给`exc`变量。

程序清单18.3包含了名为`NumberDivider`的应用程序，该应用程序从命令行接收两个整型参数，然后将它们用于除法表达式中。

该应用程序必须能够处理两个潜在的用户输入问题：

- 非数字型参数；
- 除数为0。

创建一个名为`NumberDivider`的新Java空文件，然后输入程序清单18.3中的所有文本。

程序清单18.3 `NumberDivider.java`的源代码

```
1: package com.java24hours;
2:
3: public class NumberDivider {
4:     public static void main(String[] arguments) {
5:         if (arguments.length == 2) {
6:             int result = 0;
7:             try {
8:                 result = Integer.parseInt(arguments[0]) /
9:                     Integer.parseInt(arguments[1]);
10:                 System.out.println(arguments[0] + " divided
11: by " +
12:                                     arguments[1] + " equals " + result);
13:             } catch (NumberFormatException e) {
14:                 System.out.println("Both arguments must be
15: numbers.");
16:             } catch (ArithmeticException e) {
17:                 System.out.println("You cannot divide by
18: zero.");
19:             }
20:         }
21:     }
22: }
```

使用命令行参数来指定该应用程序的两个参数。你可以将其参数指定为整数、浮点数和非数字型参数。

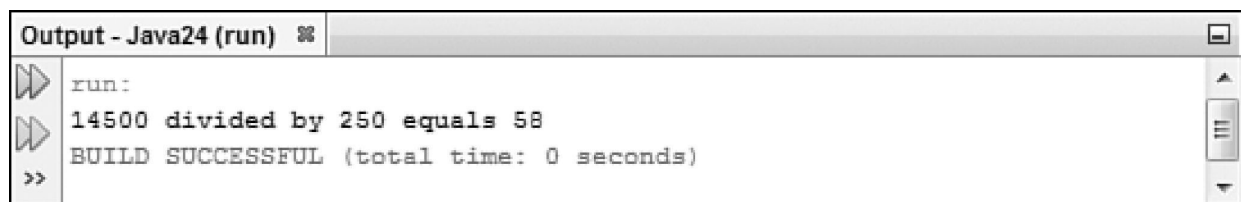
第5行的if语句检查是否给应用程序传递了两个参数，如果不是，程序将退出且不显示任何信息。

应用程序NumberDivider执行整数除法运算，因此结果也是整数。在整数除法中，5除以2等于2，而不是2.5。

如果使用浮点数或非数字作为参数，第8～9行将抛出NumberFormatException异常，并被第14～15行捕获。

如果使用整数作为第一个参数，使用0作为第二个参数，第8～9行将抛出Arithmetic Exception异常，并被第14～15行捕获。

成功运行该程序后的输出如图18.3所示。



```
Output - Java24 (run)
run:
14500 divided by 250 equals 58
BUILD SUCCESSFUL (total time: 0 seconds)
```

图18.3 NumberDivider应用程序的输出

18.1.3 出现异常后进行处理

使用try-catch块处理多种异常时，有时希望不管是否发生异常，程序都在块的后面执行某种操作。

为此，可以使用try-catch-finally块，其格式如下：

```
try {  
    // statements that might cause the exception  
} catch (Exception e) {  
    // what to do when the exception occurs  
} finally {  
    // statements to execute no matter what  
}
```

其中，finally部分的语句将在其他语句执行后执行，而不管是否发生异常。

在从磁盘文件中读取数据的程序中，会用到该功能，这将在第21章介绍。访问数据时可能发生多种异常：文件不存在、磁盘错误等。如果读取磁盘的语句在try部分，并在catch部分处理错误，可以在finally部分关闭文件。这可以确保无论读取文件是否发生异常，都将关闭文件。

18.1.4 抛出异常

调用另一个类的方法时，那个类可以通过抛出异常来控制如何使用该方法。

使用Java类库中的类时，编译器经常会显示下面这样的消息：

```
Output ▼  
NetReader.java:14: unreported exception  
java.net.MalformedURLException;  
must be caught or declared to be thrown
```

看到包含“must be caught or declared to be thrown”这样的错误消息时，表明你试图使用的方法抛出了异常。

调用这些方法的任何类，例如你编写的应用程序，必须做下面的工作之一：

- 使用try-catch块处理异常；
- 抛出异常；
- 先用try-catch块处理异常，然后再抛出异常。

至此，你看到了如何处理异常。如果想先处理异常再抛出它，可使用关键字throw和要抛出的异常对象。

下面的语句在catch块中处理错误NumberFormatException，然后再抛出它：

```
float principal;  
try {  
    principal = Float.parseFloat(loanText) * 1.1F;  
} catch (NumberFormatException e) {  
    System.out.println(arguments[0] + " is not a number.");  
    throw e;  
}
```

下面这种改写的代码可以处理try块语句中生成的所有异常，并在处理完毕后再抛出它：

```
float principal;  
try {  
    principal = Float.parseFloat(loanText) * 1.1F;  
} catch (Exception e) {  
    System.out.println("Error " + e.getMessage());  
    throw e;  
}
```

Exception是所有异常子类的父类。这个catch语句可以捕获Exception类，以及类层次结构中位于Exception类下的所有子类。

当使用throw抛出一个异常时，通常意味着没有完成处理异常需要完成的所有工作。

一个采用这种方式很有用的情形是：一个CreditCardChecker应用程序验证信用卡支付，它使用了一个名为CheckDatabase的类，该类完成如下工作：

- 连接到信用卡贷方的计算机；
- 询问该计算机，消费者的信用卡号是否有效；
- 询问该计算机，消费者是否有足够的信用。

当CheckDatabase类执行其工作时，如果信用卡贷方的计算机根本不应答连接请求将如何呢？这种错误正是try-catch块要解决的，在CheckDatabase类使用它来处理连接错误。

如果CheckDatabase类自己处理这种错误，CreditCardChecker应用程序将根本不知道发生了异常。这不是好主意—应用程序应知道不能建立连接，以便向使用应用程序的用户报告。

通知应用程序CreditCardChecker的一种方法是，在CheckDatabase类中使用catch块捕获异常，然后使用throw语句抛出它。异常将在CheckDatabase中抛出，CheckDatabase必须像处理其他异常一样处理它。

出现错误或其他不正常情况时，异常处理是类之间相互进行通信的一种方式。

当在捕获父类异常（比如Exception）的catch块中使用throw时，将会抛出该父类的异常。这样将无法获悉所发生错误的详情，因为子类异常（比如NumberFormatException）提供的信息要比Exception类更为详细。

Java提供了一种方法来记录异常的详情：catch语句中的final关键字。

```
try {
    principal = Float.parseFloat(loanText) * 1.1F;
} catch (final Exception e) {
    System.out.println("Error " + e.getMessage());
    throw e;
}
```


`catch`语句中的`final`关键字会导致`throw`语句在执行时，表现出不同的行为，即抛出所捕获的特定异常类。

18.1.5 忽略异常

本章将介绍的最后一种技术是如何完全忽略异常。在类中，可以在方法的定义中使用关键字`throw`，让方法忽略异常。

下面的方法抛出`MalformedURLException`，这是在Java程序中使用网络地址时可能发生的一种错误：

```
public void loadURL(String address) throws MalformedURLException {  
    URL page = new URL(address);  
    // code to load web page  
}
```

其中的第二条语句`URL page = new URL(address);`创建一个`URL`对象，该对象表示一个网络地址。`URL`类的构造函数抛出`MalformedURLException`，指出使用的地址无效，因此无法创建这样的对象。当你试图打开一个去往该`URL`的连接时，下面的语句将抛出这样的异常：

```
URL source = new URL("http:www.java24hours.com");
```

字符串`http:www.java24hours.com`不是有效的URL，因为冒号后少一些符号：两个反斜线（`//`）。

由于`loadURL()`方法被声明为抛出`MalformedURLException`错误，因此不用在方法中处理它们。这种异常将由调用`loadURL()`方法的方法负责处理。

18.1.6 不需要捕获的异常

尽管本章演示了一些需要使用`try-catch`来捕获的异常，以及使用`throws`语句声明为要抛出的异常，但是还有另外一种异常。

在Java程序中发生的一些异常没有必要进行处理。当编译器检测到要忽略的异常时，它不会停止运行。这些异常称为未检查（`unchecked`）异常，而其他的则称为已检查（`checked`）异常。

未检查异常是`java.lang`包中`RuntimeException`的所有子类。未检查异常的一个常见例子是`IndexOutOfBoundsException`，它指明用来访问数组、字符串或数组列表的索引超出了边界范围。如果一个数组有5个元素，而你尝试读取第10个元素，则引发这类异常。

另外一个`NullPointerException`，当使用了一个没有值的对象时，将发生该异常。在将对象变量赋值给一个对象之前，它的值为`null`。当有些方法无法返回对象时，也将返回`null`。如果语句错误地认为对象有一个值，这将发生`NullPointerException`异常。

这些错误都是程序员应该在代码中竭力避免的，而不是需要进行异常处理。如果你编写的程序访问了一个界外的数组元素，需要修改

代码然后重新编译。如果你需要一个对象，而且其值为`null`，则在使用它之前需要在`if`条件语句中进行检测。

Java中的未检查异常可以通过编写良好的代码来避免，也可以一直引发这样的异常，但是随时捕获这样的异常会让程序变得复杂。在一个程序中，调用对象的方法的每一条语句都可以引发一个`NullPointerException`。

当然，不能因为异常能被忽略就觉得应该忽略它。你仍然可以使用`try`、`catch`和`throws`来捕获未检测异常。

18.2 抛出和捕获异常

接下来创建一个类，该类使用异常将发生的错误告诉另一个类。

该类为`HomePage`，代表网上的个人主页面。`PageCatalog`是一个对这些页面进行分类的应用程序。

在一个分新的Java文件中输入程序清单18.4中的完整文本，将其命名为`HomePage`。

程序清单18.4 `HomePage.java`的完整源代码

```
1: package com.java24hours;
2:
3: import java.net.*;
4:
5: public class HomePage {
6:     String owner;
7:     URL address;
8:     String category = "none";
9: }
```

```
10:     public HomePage(String inOwner, String inAddress)
11:         throws MalformedURLException {
12:
13:         owner = inOwner;
14:         address = new URL(inAddress);
15:     }
16:
17:     public HomePage(String inOwner, String inAddress, String
18:         ➔ inCategory)
19:         throws MalformedURLException {
20:         this(inOwner, inAddress);
21:         category = inCategory;
22:     }
23: }
```

你可以在其他程序中使用这个**HomePage**类。这个类代表个人的Web页面，它包含3个实例变量：**address**（代表页面地址的URL对象）、**owner**（代表页面的主人）和**category**（描述页面主要内容的注释）。

与创建URL对象的其他类一样，**HonePage**也必须在try-catch块中处理**MalformedURLException**错误或声明忽略这些错误。

这个类采用后一种方法，如第10~11行和第17~18行所示。通过在两个构造函数中使用**throw**语句，**HomePage**避免了处理**MalformedURLException**错误。

为了创建一个使用**HomePage**类的应用程序，回到**NetBeans**并创建一个名为**PageCatalog**的Java空文件，然后输入程序清单18.5中的文本。

程序清单18.5 **PageCatalog.java**的完整源代码

```

1: package com.java24hours;
2:
3: import java.net.*;
4:
5: public class PageCatalog {
6:     public static void main(String[] arguments) {
7:         HomePage[] catalog = new HomePage[5];
8:         try {
9:             catalog[0] = new HomePage("Mark Evanier",
10:                "http://www.newsfromme.com", "comic books");
11:             catalog[1] = new HomePage("Jeff Rients",
12:                "http://jrients.blogspot.com", "gaming");
13:             catalog[2] = new HomePage("Rogers Cadenhead",
14:                "http://workbench.cadenhead.org",
15:                "programming");
16:             catalog[3] = new HomePage("Juan Cole",
17:                "http://www.juancole.com", "politics");
18:             catalog[4] = new HomePage("Rafe Colburn",
19:                "www.rc3.org");
20:             for (int i = 0; i < catalog.length; i++) {
21:                 System.out.println(catalog[i].owner + ": " +
22:                    catalog[i].address + " -- " +
23:                    catalog[i].category);
24:             }
25:         } catch (MalformedURLException e) {
26:             System.out.println("Error: " + e.getMessage());
27:         }
28:     }
}

```

运行编译后的程序时，将显示如图18.4所示的输出。

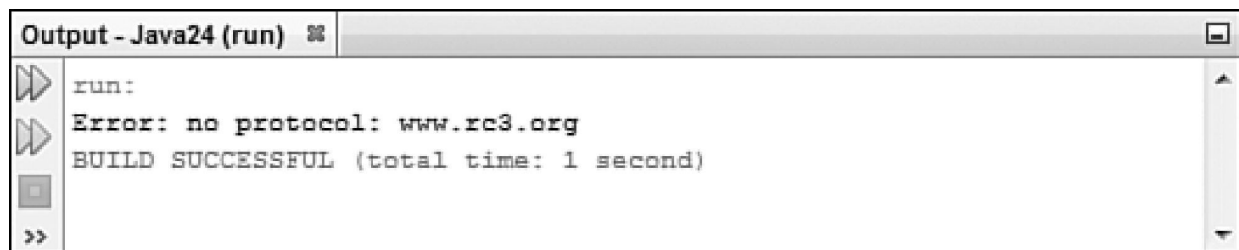


图18.4 PageCatalog应用程序的错误输出

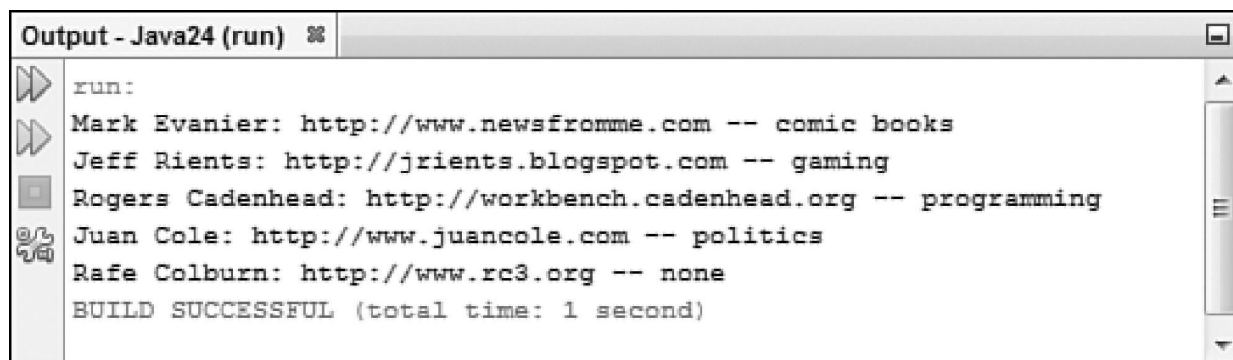
应用程序PagaCatalog创建一个HomePage对象数组，然后显示该数组的内容。创建每个HomePage对象时最多使用3个参数：

- 页面主人的姓名；
- 页面地址（String而不是URL）；
- 页面分类。

第3个参数是可选的，第17~18行没有使用。

HomePage类的构造函数收到不能转换为有效URL对象的字符串时，将抛出MalformedURLException错误。这些异常在PagaCatalog应用程序中使用来try-catch块来处理。

要修复导致“no protocol”错误的问题，编辑第18行，使字符串以http://打头，就像第9~16行的网络地址一样。当再次运行该程序时，其输出如图18.5所示。



```
Output - Java24 (run) %
run:
Mark Evanier: http://www.newsfromme.com -- comic books
Jeff Rients: http://jrients.blogspot.com -- gaming
Rogers Cadenhead: http://workbench.cadenhead.org -- programming
Juan Cole: http://www.juancole.com -- politics
Rafe Colburn: http://www.rc3.org -- none
BUILD SUCCESSFUL (total time: 1 second)
```

图18.5 PageCatalog应用程序的正确输出

18.3 总结

使用Java异常处理技术后，希望有关错误的主题比要本章开始时更受读者的欢迎。

可以使用这种技术做很多工作：

- 捕获异常并处理它；
- 忽略异常，将其留给另一个类或解释器去处理；
- 在同一个try-catch块中捕获多种异常；
- 抛出自己的异常。

通过在Java程序中管理异常，可使程序更可靠、更通用、更易于使用，因为别人在使用你的软件时，不会显示晦涩的错误消息。

18.4 问与答

问：能够创建自己的异常吗？

答： 可以通过创建现有异常（如Exception，它是所有异常的超类）的子类来创建自己的异常。在Exception的子类中，可能需要覆盖两个方法：不接受任何参数的Exception()方法和接受一个String参数的Exception()方法。在后一个方法中，字符串参数应是一条消息，它描述发生的错误。

问：除异常外，本章为什么没有描述如何抛出和捕获错误？

答： Java将问题分为错误和异常两类，因为它们的严重程度不同。异常不那么严重，因此应在程序中使用try-catch或throw来处理它们；而错误更严重，无法在程序中进行妥善处理。

有关错误的两个例子是栈溢出和内存耗尽，这些错误可能导致Java解释器崩溃，而且在解释器运行时，没有办法在程序中修复这种错误。

18.5 测验

本章充斥“错误”一词，看看能否不出任何错误地回答下列问题。

18.5.1 问题

1. 单条catch语句能够处理多少种异常？
 - a. 一种。
 - b. 多种不同的异常。
 - c. 我不想回答这个问题。
2. finally部分中的语句在什么情况下运行？
 - a. 出现异常且try-catch块结束后。
 - b. 不出现异常且try-catch块结束后。
 - c. 两者都是。

18.5.2 答案

1. b. catch语句中的Exception对象能够处理它自己的类以及它的超类的所有异常。

2. c. `finally`部分中的语句总是在`try-catch`块执行完毕后执行，而不管是否发生异常。

18.6 练习

要检测你是否是一名优秀的Java程序员，尽量少犯错误地完成下列练习。

- 修改应用程序`NumberDivider`，使其抛出它能够捕获的任何异常。然后运行程序，看看发生的情况。
- 第15章创建的`LottoEvent`类有一个`try-catch`块，根据这个块创建自己的`Sleep`类，它处理`InterruptedException`异常，这样其他类（如`LottoEvent`）就不需要处理这种异常了。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第19章 创建线程程序

本章介绍如下内容：

- 在程序中使用接口；
- 创建线程；
- 启动、终止和暂停线程；
- 捕获错误。

一个常用于描述每天紧张忙碌的生活节奏的计算机术语是“多任务”，意思是同时做多件事，如一边浏览Web站点，一边参加电话会议，同时还做压腿练习。多线程计算机是同时可以运行多个程序的计算机。

Java语言的一个复杂特性是，能够编写具有多任务功能的程序，Java的该功能是通过称为线程的对象实现的。

19.1 线程

在Java程序中，计算机同时处理的每一个任务称为“线程”，所有的任务称为“多线程”。线程在动画和很多其他程序中很有用。

线程是一种组织程序的方式，使其能够同时做多项工作。必须同时执行的每项任务都运行在自己的线程中，这通常是通过将每项任务作为独立的类来实现的。

线程用Thread类和Runnable接口表示，它们都位于java.lang包中。由于它们都位于这个包中，因此在程序不用使用import语句就能使用它们。

Thread类的一种最简单的用途是降低程序完成工作的速度。

19.1.1 降低程序的速度

Thread类有一个sleep()方法，可在任何程序中调用它让程序停止运行一段时间。在动画

程序中经常使用这种技术，以防止图像显示的速度超过Java解释器的处理能力。

要使用sleep()方法，可调用Thread.sleep()方法并使用要暂停的毫秒数作参数，如下面的语句所示：

```
Thread.sleep(5000);
```

上述语句让Java虚拟机暂停5秒再继续运行。如果由于某种原因JVM不能暂停那么长时间，sleep()方法将抛出一个InterruptedException异常。

由于可能会抛出上面提到的异常，因此，在使用sleep()方法时必须采取某种方式处理这种异常。一种解决办法是将Thread.sleep()语句放在try-catch块中：

```
try {  
    Thread.sleep(5000);  
} catch (InterruptedException e) {  
    // wake up early  
}
```

要让Java程序同时处理多项任务，必须将程序组织成线程。根据需要，程序可以有任意多个线程，它们将同时运行，而且不会相互影响。

19.1.2 创建线程

可作为线程运行的Java类常被称为“可运行类或线程化类”。虽然使用线程可让程序暂停几秒再执行，但通常因为相反的原因而使用它们：提高程序的运行速度。如果将耗时的任务放在独立的线程中，程序的其他部分运行起来将更快，这通常用于防止任务降低程序的图形用户界面的响应速度。

例如，如果编写了一个应用程序，它从磁盘加载股价数据并生成统计数据，则最耗时的任务是从磁盘加载数据。如果不在应用程序中使用线程，加载数据时程序的界面响应将很慢。这可能让用户失去耐心。

有两种方法可将任务放在线程中：

- 将任务放在实现了Runnable接口的类中；
- 将任务放在Thread的子类中。

要支持**Runnable**接口，可在创建类时使用关键字**implements**，如下例所示：

```
public class LoadStocks implements Runnable {  
    // body of the class  
}
```

当类实现了接口后，除了自己的方法外，它还将支持其他的行为。

实现了**Runnable**接口的类必须包含**run()**方法，该方法的结构如下：

```
public void run() {  
    // body of the method  
}
```

run()方法应完成线程要完成的任务。在股票分析示例中，**run()**方法应包含从磁盘加载数据以及根据这些数据生成统计信息的语句。

当线程应用程序运行时，不会自动执行**run()**方法中的语句。在Java中可以启动和终止线程，要运行线程，必须做下面两项工作：

- 通过调用**Thread**构造函数创建线程化类的对象；
- 通过调用**start()**方法启动线程。

构造函数Thread接受一个参数——包含线程的run()方法的对象。通常使用this作为参数，this表明当前类包含run()方法。

程序清单19.1所示的Java应用程序在文本区域显示一系列素数。在NetBeans创建一个名为Prime Finder的Java空文件，然后输入程序清单19.1中的所有文本，并保存。

程序清单19.1 PrimeFinder.java程序的源代码

```
1: package com.java24hours;
2:
3: import java.awt.*;
4: import javax.swing.*;
5: import java.awt.event.*;
6:
7: public class PrimeFinder extends JFrame implements Runnable,
   ➤ ActionListener {
8:     Thread go;
9:     JLabel howManyLabel;
10:    JTextField howMany;
11:    JButton display;
12:    JTextArea primes;
13:
14:    public PrimeFinder() {
15:        super("Find Prime Numbers");
16:        setLookAndFeel();
17:        setSize(400, 300);
18:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19:        BorderLayout bord = new BorderLayout();
20:        setLayout(bord);
21:
22:        howManyLabel = new JLabel("Quantity: ");
23:        howMany = new JTextField("400", 10);
24:        display = new JButton("Display primes");
25:        primes = new JTextArea(8, 40);
26:
27:        display.addActionListener(this);
28:        JPanel topPanel = new JPanel();
29:        topPanel.add(howManyLabel);
30:        topPanel.add(howMany);
31:        topPanel.add(display);
32:        add(topPanel, BorderLayout.NORTH);
33:    }
```

```

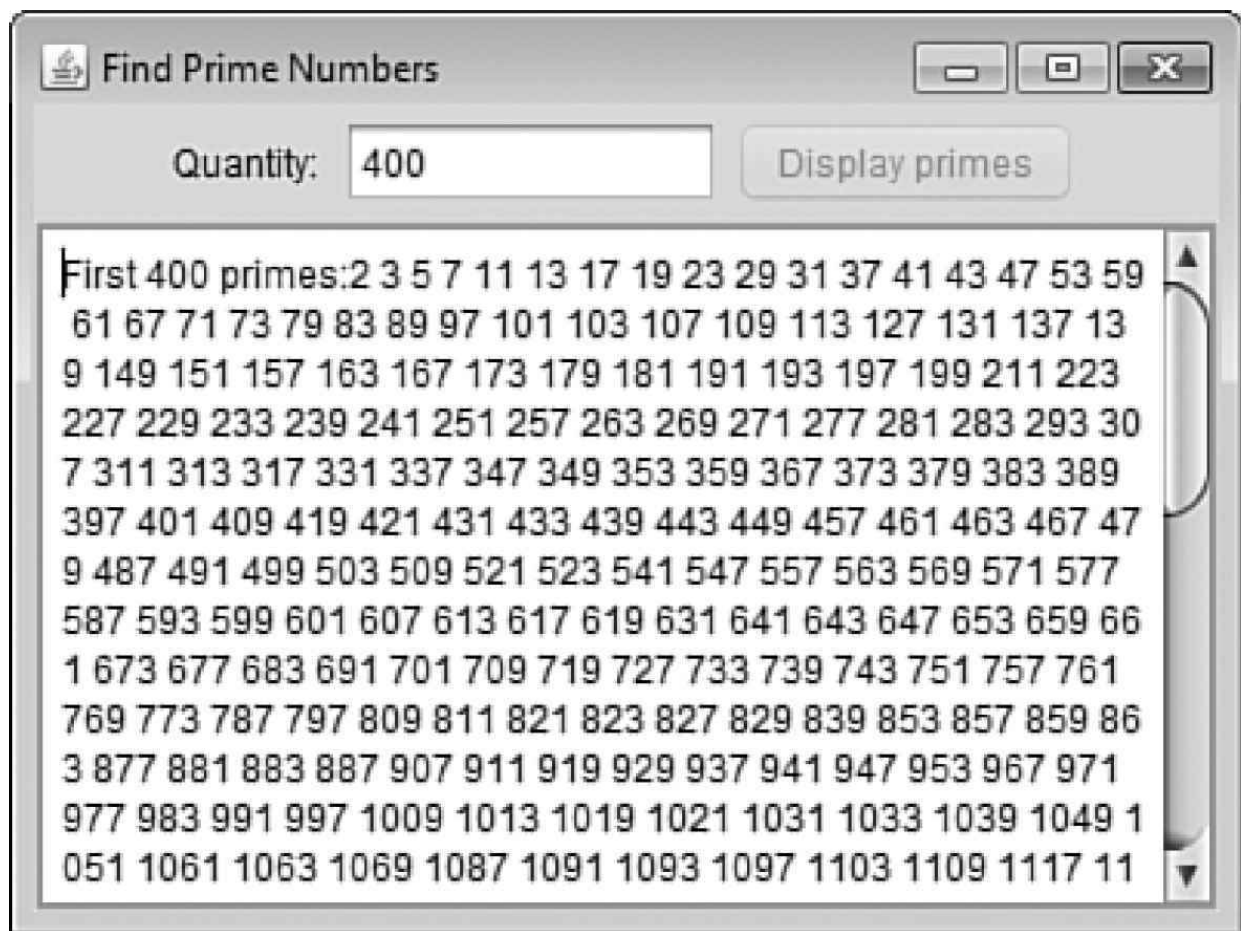
34:         primes.setLineWrap(true);
35:         JScrollPane textPane = new JScrollPane(primes);
36:         add(textPane, BorderLayout.CENTER);
37:
38:         setVisible(true);
39:     }
40:
41:     public void actionPerformed(ActionEvent event) {
42:         display.setEnabled(false);
43:         if (go == null) {
44:             go = new Thread(this);
45:             go.start();
46:         }
47:     }
48:
49:     public void run() {
50:         int quantity = Integer.parseInt(howMany.getText());
51:         int numPrimes = 0;
52:         // candidate: the number that might be prime
53:         int candidate = 2;
54:         primes.append("First " + quantity + " primes:");
55:         while (numPrimes < quantity) {
56:             if (isPrime(candidate)) {
57:                 primes.append(candidate + " ");
58:                 numPrimes++;
59:             }
60:             candidate++;
61:         }
62:     }
63:
64:     public static boolean isPrime(int checkNumber) {
65:         double root = Math.sqrt(checkNumber);
66:         for (int i = 2; i <= root; i++) {
67:             if (checkNumber % i == 0) {
68:                 return false;
69:             }
70:         }
71:         return true;
72:     }
73:
74:     private void setLookAndFeel() {
75:         try {
76:             UIManager.setLookAndFeel(
77: "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
78:             );
79:         } catch (Exception exc) {
80:             // ignore error
81:         }
82:     }
83:

```

```
84:    public static void main(String[] arguments) {  
85:        new PrimeFinder();  
86:    }  
87: }
```

应用程序PrimeFinder显示一个文本框、一个Display primes按钮和一个文本区域，如图19.1所示。

该应用程序中的大部分语句用于创建图形用户界面和显示一系列素数。下面是该程序中用于实现线程的语句。



- 第7行：将Runnable接口应用到PrimeFinder类。
- 第8行：创建一个Thread对象变量，名字为go，但是没有为其赋值。
- 第43～46行：如果对象变量go的值为null，表示该线程还没有创建，因此创建一个新的Thread对象并将其存储到go变量中。通过调用线程的start()方法启动线程，这导致PrimeFinder类中的run()方法被调用。
- 第49～62行：run()方法从2开始寻找一系列素数，并通过调用文本区域组件primes的append()方法显示每个素数。素数个数由howMany文本框中的值决定。

当创建一个PrimeFinder对象来执行程序时，该程序中的main()方法中还有一些不同。下面是语句：

```
new PrimeFinder();
```

通常情况下，你可能希望看到该对象被赋值给一个变量，如下所示：

```
PrimeFinder frame = new PrimeFinder();
```

然而，因为这里没有必要再次引用那个对象，所以就没有必要将其存储到一个变量中。调用new来创建对象时，将运行该程序。

只在需要时将对象存储到变量中是一种很好的Java编程习惯。

19.2 使用线程

可以通过调用`start()`方法启动线程，因此读者可能认为也有一个`stop()`方法用于终止线程。

尽管在Java的`Thread`类中包含`stop()`方法，但该方法已经被废弃。在Java中，被废弃的元素是指已被更好的东西取代的类、接口、方法或变量。

Watch Out!

警告：

读者应该重视有关废弃使用的警告。Oracle公司之所以废弃`stop()`方法，是因为它可能给Java解释器中运行的其他线程带来问题。线程类的`resume()`方法和`suspend()`方法也已经被废弃。

下一个项目将演示如何终止线程。读者将编写的程序将在一个列表中循环，该列表包含网站名称及其网址。

每一个页面的标题和地址将循环显示。用户可以单击应用程序图形用户界面中的按钮来访问当前显示的网站。该程序按顺序显示每个网站的信息。由于时间因素，使用线程来控制程序是最好的选择。

这次不是首先在NetBeans的源代码编辑器中输入程序然后再介绍它。本章结束时读者将有机会输入LinkRotator应用程序的全部代码，

在此之前，将描述该程序的每个组成部分。

19.2.1 声明类

在这个程序中，首先需要使用import语句导入java.awt、java.net、java.applet、java.awt.event和javax.swing中的一些类。

使用import语句导入一些类后，就可以使用下面的语句创建应用程序：

```
public class LinkRotator extends JFrame
    implements Runnable, ActionListener {
```

与你之前开发的图形用户界面应用程序一样，该语句将LinkRotator类创建为JApplet的子类，并指出它支持两个接口：Runnable和ActionListener。通过实现Runnable接口，将能够在该程序中使用run()方法来启动线程；ActionListener接口让程序能够响单击鼠标的操作。

19.2.2 创建变量

在LinkRotator类中，首先要创建该类的变量和对象。创建两个包含6个元素的数组：一个名为pageTitle的String对象数组和一个名为pageLink的URL对象数组：

```
String[] pageTitle = new String[6];
URI[] pageLink = new URI[6];
```

`pageTitle`数组用于存储要显示的6个网站的标题。`URL`对象存储网站地址，`URL`有记录网站地址所需的行为和属性。

最后3项工作是创建一个整型变量、一个`Thread`对象，以及一个用户界面标签：

```
int current = 0;
Thread runner;
JLabel siteLabel = new JLabel();
```

变量`current`用于记录当前显示的网址，以便能够在网站之间循环。`Thread`对象`runner`是该程序运行的线程。可调用`runner`对象的方法来启动、终止和暂停程序的运行。

19.3 构造函数

该程序在运行时，将自动执行程序的构造函数。该方法用于给数组`pageTitle`和`pageLink`赋值。它还可以用来创建显示在用户界面中的一个可单击按钮。该方法由下面的语句组成：

```
pageTitle = new String[] {
    "Oracle's Java site",
    "Cafe au Lait",
    "JavaWorld",
    "Java in 24 Hours",
    "Sams Publishing",
    "Workbench"
}
```

```
};  
pageLink[0] = getURI("http://www.oracle.com/technetwork/java");  
pageLink[1] = getURI("http://www.ibiblio.org/javafaq");  
pageLink[2] = getURI("http://www.javaworld.com");  
pageLink[3] = getURI("http://www.java24hours.com");  
pageLink[4] = getURI("http://www.sampublishing.com");  
pageLink[5] = getURI("http://workbench.cadenhead.org");  
Button visitButton = new Button("Visit Site");  
goButton.addActionListener(this);  
add(visitButton);
```

每一个网站的标题存储到pageTitle数组的6个元素中，这是用6个字符串来初始化的。PageLink数组的元素由getURL()方法返回的值来确定，getURL()方法此时还没有创建。

init()方法中的最后3条语句用于创建一个按钮并将添加在应用程序中。该按钮的标签为文本“Visit Site”。

19.4 在创建URL时捕获错误

创建URL对象时，必须确保用于设置网址的文本为有效格式。http://workbench.cadenhead.org和http://www.sampublishing.com都是有效的，但是http:www.javaworld.com不是有效的，因为少了//。

getURL (String) 方法接收一个网络地址作为参数，然后返回一个表示该地址的URL对象。如果作为参数的字符串不是有效的地址，则返回null。

```
URI getURI(String urlText) {  
    URI pageURI = null;  
    try {
```

```
        pageURI = new URI(urlText);
    } catch (URISyntaxException m) {
        // do nothing
    }
    return pageURI;
}
```

在创建URL对象时，**try-catch**语句块用来处理发生的任何**MalformedURLException**错误。由于在该错误抛出时，什么都没有发生，因此catch语句块只包含注释行。

19.5 启动线程

在这个程序中，**runner**线程将在**start()**方法被调用时启动。

start()方法作为构造函数的最后一条语句被调用。下面是**start()**方法：

```
public void start() {
    if (runner == null) {
        runner = new Thread(this);
        runner.start();
    }
}
```

如果**runner**线程还没有启动，该方法将启动它。

语句`runner = new Thread (this);`使用一个参数（关键字`this`）创建一个新的`Thread`对象，关键字`this`引用的是程序自身，这使该程序作为类运行在`runner`线程中。

语句`runner.start();`导致线程开始运行。线程开始后，将调用线程的`run()`方法。由于`runner`线程是`LinkRotator`程序本身，因此将调用该程序的`run()`方法。

19.5.1 运行线程

线程的主要工作是在`run()`方法中完成的。在`LinkRotator`程序中，`run()`方法如下：

```
public void run() {
    Thread thisThread = Thread.currentThread();
    while (runner == thisThread) {
        current++;
        if (current > 5) {
            current = 0;
        }
        siteLabel.setText(pageTitle[current]);
        repaint();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            // do nothing
        }
    }
}
```

`run()`方法做的第一件事情是创建一个名为`thisThread`的`Thread`对象，使用`Thread`类的类方法`currentThread()`给`thisThread`对象赋值，`currentThread()`方法跟踪当前运行的线程。

该方法的所有语句都位于while循环中，该循环比较runner对象和thisThread对象。这两个对象都是线程，而且只要引用的是同一个对象，while循环将一直执行下去。该循环中没有导致runner和thisThread对象的值不同的语句，因此循环将不断进行下去，直到在循环外修改了其中一个线程对象。

run()方法首先使用repaint();语句，接下来将current变量的值加1，如果current大于5，则将其重置为0。变量current用作pageTitle字符串数组的索引，而且标题被设置为siteLabel用户界面组件的文本。

方法run()包括另外一个try-catch语句，负责处理可能发生的错误。语句Thread.sleep(1000);让线程暂停1秒。该语句导致线程等待一段时间，让用户就有足够的时间阅读网站的名称和网址。catch语句处理调用Thread.sleep()方法时可能发生的InterruptedException错误，如果在线程休眠时将其中断，将发生这种错误。

19.6 处理鼠标单击

LinkRotator程序需要完成的最后一项工作是事件处理：每当用户单击Visit Site按钮，应用程序应该在Web浏览器中打开列出的网站。这是使用actionPerformed()方法实现的，该方法在用户单击按钮时被调用。

下面是LinkRotator的actionPerformed()方法：

```
public void actionPerformed(ActionEvent event) {
    Desktop desktop = Desktop.getDesktop();
    if (pageLink[current] != null) {
```



```
        try {
            desktop.browse(pageLink[current]);
            runner = null;
            System.exit(0);
        } catch (IOException exc) {
            // do nothing
        }
    }
}
```

该方法要做的第一件事是创建Desktop对象。Desktop类位于java.awt包中，表示运行应用程序的计算机的桌面环境。在拥有了这个对象后，你可以使用一个“mailto:”链接启动一个电子邮件客户端，打开要在另外一个程序中编辑的文件，打印文件，以及让其他程序执行任务。

在这里，Desktop对象使用计算机的默认Web浏览器打开一个网页。

方法browse(URI)在浏览器中载入指定的网页地址。如果pageLink[current]是一个有效的地址，则browse()会请求浏览器载入该网页。

19.7 循环显示链接

现在准备创建LinkRotator程序并进行测试。创建一个名为LinkRotator的Java空文件，然后输入程序清单19.2所示的内容。

程序清单19.2 LinkRotator.java程序的完整源代码

```
1: package com.java24hours;
```

```

2:
3: import java.awt.*;
4: import java.awt.event.*;
5: import java.io.*;
6: import javax.swing.*;
7: import java.net.*;
8:
9: public class LinkRotator extends JFrame
10:     implements Runnable, ActionListener {
11:
12:     String[] pageTitle = new String[6];
13:     URI[] pageLink = new URI[6];
14:     int current = 0;
15:     Thread runner;
16:     JLabel siteLabel = new JLabel();
17:
18:     public LinkRotator() {
19:         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20:         setSize(300, 100);
21:         FlowLayout flo = new FlowLayout();
22:         setLayout(flo);
23:         add(siteLabel);
24:         pageTitle = new String[] {
25:             "Oracle's Java site",
26:             "Cafe au Lait",
27:             "JavaWorld",
28:             "Java in 24 Hours",
29:             "Sams Publishing",
30:             "Workbench"
31:         };
32:         pageLink[0] = getURI("http://www.oracle.com/technetwork/
           ➡ java");
33:         pageLink[1] = getURI("http://www.ibiblio.org/javafaq");
34:         pageLink[2] = getURI("http://www.javaworld.com");
35:         pageLink[3] = getURI("http://www.java24hours.com");
36:         pageLink[4] = getURI("http://www.sampublishing.com");
37:         pageLink[5] = getURI("http://workbench.cadenhead.org");
38:         Button visitButton = new Button("Visit Site");
39:         visitButton.addActionListener(this);
40:         add(visitButton);
41:         setVisible(true);
42:         start();
43:     }
44:
45:     private URI getURI(String urlText) {
46:         URI pageURI = null;
47:         try {
48:             pageURI = new URI(urlText);
49:         } catch (URISyntaxException ex) {
50:             // do nothing
51:         }

```

```
52:         return pageURI;
53:     }
54:
55:     public void start() {
56:         if (runner == null) {
57:             runner = new Thread(this);
58:             runner.start();
59:         }
60:     }
61:
62:     public void run() {
63:         Thread thisThread = Thread.currentThread();
64:         while (runner == thisThread) {
65:             current++;
66:             if (current > 5) {
67:                 current = 0;
68:             }
69:             siteLabel.setText(pageTitle[current]);
70:             repaint();
71:             try {
72:                 Thread.sleep(1000);
73:             } catch (InterruptedException exc) {
74:                 // do nothing
75:             }
76:         }
77:     }
78:
79:     public void actionPerformed(ActionEvent event) {
80:         Desktop desktop = Desktop.getDesktop();
81:         if (pageLink[current] != null) {
82:             try {
83:                 desktop.browse(pageLink[current]);
84:                 runner = null;
85:                 System.exit(0);
86:             } catch (IOException exc) {
87:                 // do nothing
88:             }
89:         }
90:     }
91:
92:     public static void main(String[] arguments) {
93:         new LinkRotator();
94:     }
95: }
```

图19.2所示为在计算机桌面中打开了两个窗口。在椭圆形的那个小窗口是正在运行的LinkRotator应用程序。较大的窗口是程序打开链接后显示的窗口：这里是本书出版社Sams的网页。

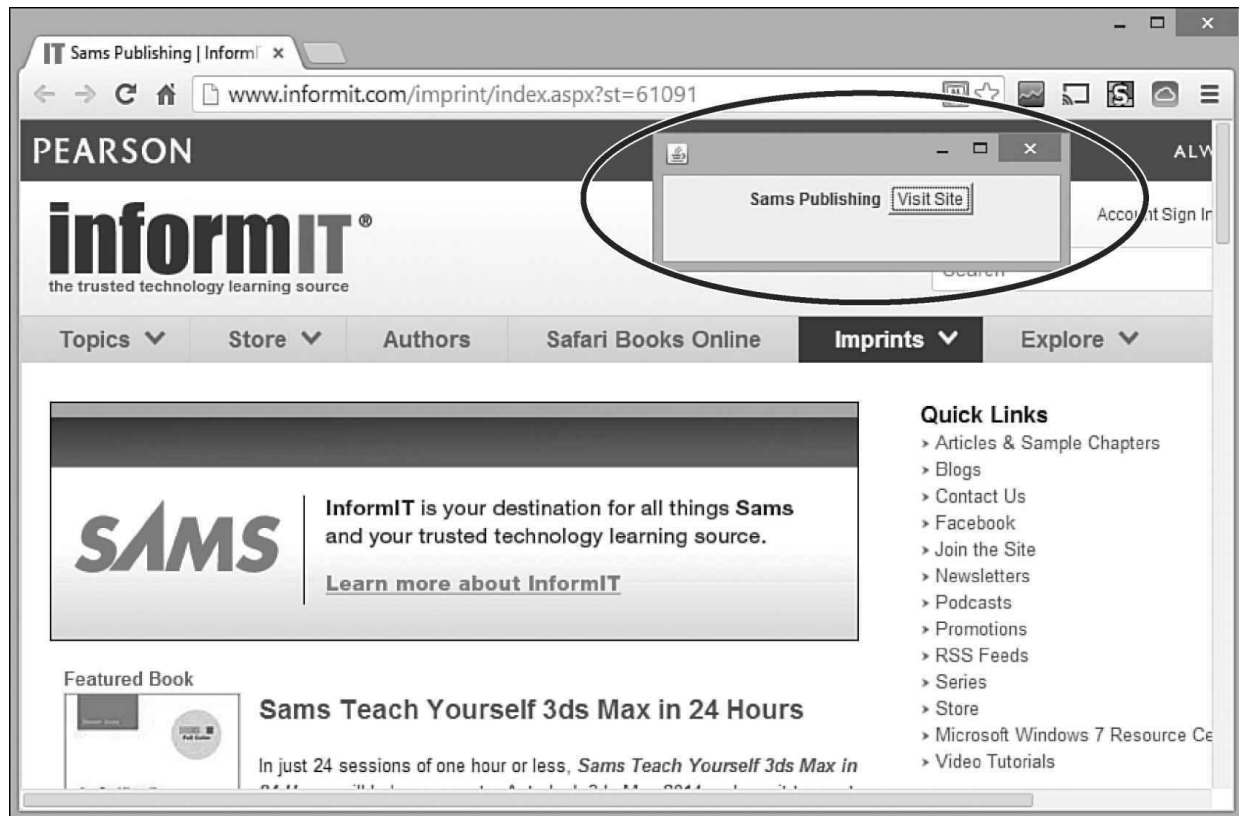


图19.2 在应用程序中循环显示链接

19.7.1 停止线程

LinkRotator应用程序没有方法来停止线程，但是我们可以很容易地来实现该功能。可以调用下面这个方法来终止线程的执行：

```
public void stop() {  
    if (runner != null) {  
        runner = null;  
    }  
}
```

if语句用来测试runner对象是否等于null。如果是，则没有需要停止的活跃进程。否则，语句将runner设置为null。

将runner对象设置为null值后，将使其与thisThread对象的值不在相同。此时，run()方法中的while循环将停止运行。

19.8 总结

线程是一个功能强大的概念，在Java中通过几个类和接口来实现。通过在应用程序中支持多线程，可提高程序的响应速度及其执行任务的速度。

即使读者在本章中没有学到其他东西，但至少现在知道了一个用于描述狂热生活方式的新术语，那就是多线程（multithreading）。

19.9 问与答

问：在LinkRotator 应用程序中的catch语句中不执行任何操作，这是为什么呢？

答：这取决于所捕获的错误或异常的类型。在LinkRotator应用程序中，由于你已经知道引发异常的原因是什么，因此catch语句中可以不执行任何操作，这都不会有问题。在getURL()方法中，只有当发送给该方法的URL无效，才会引发MalformedURLException异常。

19.10 测验

将你的线程放在一边，回答下列关于Java多线程的问题。

19.10.1 问题

1. 程序要使用线程，必须实现哪个接口？

- a. Runnable。
- b. Thread。
- c. 不需要接口。

2. 如果接口包含3个不同的方法，在实现了该接口的类中必须包含其中的几个方法？

- a. 一个也不需要。
- b. 所有的方法。
- c. 我知道，但是我不说。

19.10.2 答案

1. a. 必须使用关键字implements来实现Runnable。Thread用于多线程程序中，但不需要在程序开头的class语句中使用。

2. b. 接口要求实现它的类必须包括其所有方法。

19.11 练习

如果本章内容还没有让读者感觉到劳累，请完成下面的练习以提高技能。

- 创建一个新版本的**LinkRotator**应用程序，其中包含6个你最喜欢的网站。
- 在应用程序**PrimeFinder**中添加一个按钮，以便在计算素数时能停止线程。

有关为完成这些练习而编写的**Java**程序，请访问本书的配套网站 www.java24hours.com。

第20章 使用内部类和闭包

本章介绍如下内容：

- 在对象中添加内部类；
- 内部类为什么很有用；
- 创建匿名的内部类；
- 在接口中使用适配器类；
- 在Java 8中添加闭包的原因；
- 编写lambda表达式；
- 使用lambda表达式替换匿名内部类。

在Java编程语言于1995年发布之时，它的使用范围有限而且也比较容易掌握。当时在Java类库中只有大约250多个类。而且Java语言的设计初衷主要是让applet交互程序能在Web浏览器内运行。由于当时还没有其他技术能够实现这一功能，因此Java引起了巨大的轰动，并且很快就吸引了成千上万的程序员。

Java语言设计的相当成功，它很快就超越了其最初的主旨（即在Web浏览器上运行applet），演变成为一种通用的编程语言，并成为C++的竞争对手。在那时，C++是最流行、应用最广泛的编程语言。

在Java语言诞生近20年的今天，有数百万程序员使用Java进行编程。Java每发布一个新版本，它都会支持现有开发人员的同时，扩展其功能，以使用复杂的软件开发新方法。

Java 8引入了一种令人激动的新方式来编写称为闭包或lambda表达式的代码。这些表达式使得函数式编程称为可能。

在本章，你将学习lambda表达式，但是在此之前会先讲解使用lambda表达式的两个先决条件：内部类和匿名内部类。

20.1 内部类

在Java中创建类时，也就定义了类的属性和行为。属性是用来存放数据的类和实例变量，行为是使用存放的数据来执行任务的方法。

类中还可以包含由属性和行为组成的其他东西：内部类。

内部类是包含在一个封闭类中的辅助类。既然可以使用很多新类创建程序，为什么还需要内部类呢？如果你在编写一个CheckBook程序，该程序中，编写的每一个检查（**check**）都需要对象，你需要创建一个Check类。如果该程序支持按月份付款，则可以添加一个Autopayment类。

内部类则是没有必要的。但是当你知道了内部类能做的事情之后，就会发现在很多情况下，内部类会派上用场。

在Java中包含内部类的原因有3个。

1. 当辅助类只能被另外一个类使用时，就可以在那个类中定义辅助类。

2. 这可以使得辅助类能够访问`private`方法和变量；而这些方法和变量是不能被单独的类所访问的。

3. 当在另一个类中使用辅助类时，可以让辅助类距离该类足够近。

与其他类一样，可以使用`class`关键字来创建内部类，但是它是在所属类的内部声明的。通常类的内部是放置类和实例变量的地方。

下面是一个`InnerSimple`内部类的例子，它是在`Simple`类中创建的：

```
public class Simple {  
    class InnerSimple {  
        InnerSimple() {  
            System.out.println("I am an inner class!");  
        }  
    }  
  
    public Simple() {  
        // empty constructor  
    }  
  
    public static void main(String[] arguments) {  
        Simple program = new Simple();  
        Simple.InnerSimple inner = program.newInnerSimple();  
    }  
}
```

内部类的结构与其他类一样，但是位置是在封闭类的“{”和“}”内部。

创建内部类需要外部类的对象，然后这个外部类对象调用new操作符：

```
Simple.InnerSimple inner = program.newInnerSimple();
```

类名包含了外部类的名字、点号（.），然后是内部类的名字。在上面的语句中，Simple.InnerSimple是类名。

本章的第一个项目是重写第18章的应用程序PageCatalog。该应用程序需要一个HomePage辅助类。代码清单20.1中的Catalog应用程序修改了该项目，将那个单独的类使用内部类进行了替换。

在com.java24hours包中创建一个新的Java文件，命名为Catalog，然后输入程序清单20.1中的源代码。

程序清单20.1 Catalog.java的完整源代码

```
1: package com.java24hours;
2:
3: import java.net.*;
4:
5: public class Catalog {
6:     class HomePage {
7:         String owner;
8:         URL address;
9:         String category = "none";
10:
11:         public HomePage(String inOwner, String inAddress)
12:             throws MalformedURLException {
13:
14:             owner = inOwner;
15:             address = new URL(inAddress);
16:         }
```

```

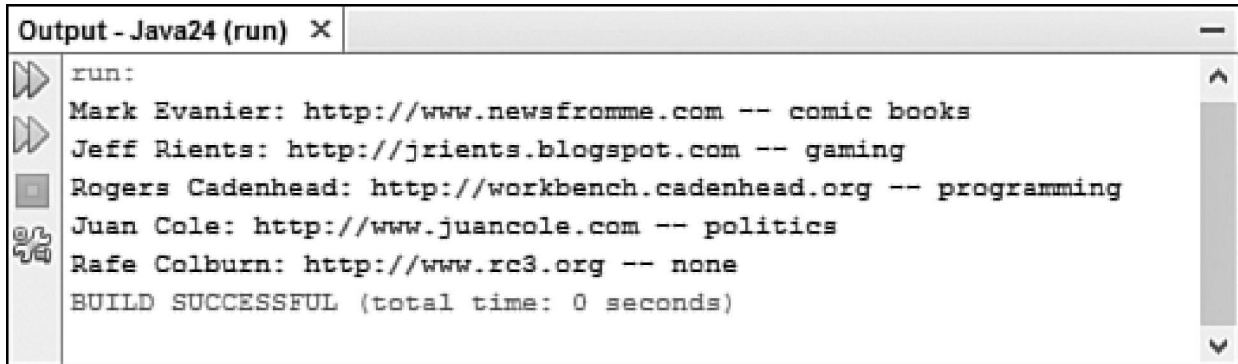
17:
18:     public HomePage(String inOwner, String inAddress,
String
    ➔ inCategory)
19:         throws MalformedURLException {
20:
21:         this(inOwner, inAddress);
22:         category = inCategory;
23:     }
24: }
25:
26: public Catalog() {
27:     Catalog.HomePage[] catalog = new Catalog.HomePage[5];
28:     try {
29:         catalog[0] = new HomePage("Mark Evanier",
30:             "http://www.newsfromme.com", "comic books");
31:         catalog[1] = new HomePage("Jeff Rients",
32:             "http://jrients.blogspot.com", "gaming");
33:         catalog[2] = new HomePage("Rogers Cadenhead",
34:             "http://workbench.cadenhead.org",
"programming");
35:         catalog[3] = new HomePage("Juan Cole",
36:             "http://www.juancole.com", "politics");
37:         catalog[4] = new HomePage("Rafe Colburn",
38:             "http://www.rc3.org");
39:         for (int i = 0; i < catalog.length; i++) {
40:             System.out.println(catalog[i].owner + ": " +
41:                 catalog[i].address + " -- " +
42:                 catalog[i].category);
43:         }
44:     } catch (MalformedURLException e) {
45:         System.out.println("Error: " + e.getMessage());
46:     }
47: }
48:
49: public static void main(String[] arguments) {
50:     new Catalog();
51: }
52: }

```

内部类是在第6～24行定义的。它有两个构造函数，在第11行的构造函数接收一个网站的主人和网站的URI作为参数；第18行的另外一个构造函数接收网站的主人、URI和分类作为参数。

Catalog类在第27行使用了这个内部类，创建了一个HomePage对象数组。内部类被称为Catalog.HomePage。

该应用程序的输出如图20.1所示。



```
Output - Java24 (run) X
run:
Mark Evanier: http://www.newsfromme.com -- comic books
Jeff Rients: http://jrients.blogspot.com -- gaming
Rogers Cadenhead: http://workbench.cadenhead.org -- programming
Juan Cole: http://www.juancole.com -- politics
Rafe Colburn: http://www.rc3.org -- none
BUILD SUCCESSFUL (total time: 0 seconds)
```

图20.1 Catalog应用程序的输出

20.1.1 匿名内部类

在Java编程中经常执行的一个任务是创建一个只使用一次且用途简单的类。一种特殊的内部类可以完美地实现该目的。

匿名内部类是没有名字，而且声明与创建是在同时进行的类。

要使用匿名内部类，要使用new关键字以及“{”和“}”内部的类定义来替换对对象的引用，其中new关键字会调用构造函数。

下面是不实用匿名内部类的代码：

```
WorkerClass worker = new WorkerClass();
Thread main = new Thread(worker);
main.start();
```

对象`worker`实现了`Runnable`接口，而且可以作为一个线程运行。

如果`WorkerClass`中的代码短小简单，而且该类只使用一次，就可以将其放到一个内部类中。下面是实现该目的的代码：

```
Thread main = new Thread(new Runnable() {  
    public void run() {  
        // thread's work to perform goes here  
    }  
});  
main.start();
```

匿名内部类已经使用下面的代码替换了`worker`对象的引用：

```
new Runnable() {  
    public void run() {  
        // thread's work to perform goes here  
    }  
}
```

这创建了一个匿名类，而且实现了`Runnable`接口并覆盖了`run()`方法。该方法内的语句可以执行类需要的任何工作。

在学习了如何编写一个匿名内部类以及了解到它为什么有用之后，读者将更容易理解匿名内部类的概念。

你在前面章节创建的项目将使用内部类来进行改善。

在第15章，你创建了一个接受键盘输入的Swing应用程序。
KeyViewer应用程序使用一个实现了KeyListener接口（应用程序自己的类）的对象来监控键盘输入。

实现了接口的类必须实现3个方法：keyTyped()、keyPressed()和keyReleased()。下面的代码是它们在程序中的设计代码。

```
public void keyTyped(KeyEvent input) {
    char key = input.getKeyChar();
    keyLabel.setText("You pressed " + key);
}

public void keyPressed(KeyEvent txt) {
    // do nothing
}

public void keyReleased(KeyEvent txt) {
    // do nothing
}
```

然后设置键盘监听器，使其使用该类来监控键盘事件：

```
keyText.addKeyListener(this);
```

借助于一个匿名内部类和java.awt.event包中的KeyAdapter包，可以有更好的方法来创建监听器，并将其添加到图形用户界面中。

KeyAdapter类实现了**KeyListener**接口，但是没有实现3个方法。这可以很容易地为键盘事件创建一个监听器，因为你可以创建一个只覆盖方法的子类，该子类为所需操作的实际执行者。

下面是**KeyViewer**应用程序中为键盘监听器的框架。

```
public class KeyViewerListener extends KeyAdapter {  
    public void keyTyped(KeyEvent input) {  
        // to do  
    }  
}
```

该监听器可以作为一个监听器来创建和设置：

```
KeyViewerListener kv1 = new KeyViewerListener();  
keyText.addKeyListener(kv1);
```

该方法需要一个单独的辅助类：**KeyViewerListener**，还需要创建该类的一个对象，并赋值给一个变量。

实现该目的另外一个方法是将监听器创建为一个匿名内部类：

```
keyText.addKeyListener(new KeyAdapter() {  
    public void keyTyped(KeyEvent input) {  
        char key = input.getKeyChar();  
        keyLabel.setText("You pressed " + key);  
    }  
});
```


该监听器是通过调用`new KeyAdapter()`，后跟类的定义来匿名创建的。这个类覆盖了`keyTyped()`方法，这样当按下一个键时，将通过调用`getChar()`来检索到，然后通过设置`keyLabel`的值进行显示，其中`keyLabel`是一个`JLabel`组件。

By the Way

注意

在`java.awt.event`包中还有一些其他适配器类，可以用来方便地实现其他监听器。`MouseAdapter`类有用于3个鼠标监听器接口的无实际功能（do-nothing）的方法，`WindowAdapter`实现了3个窗口监听器，`FocusAdapter`实现了`FocusListener`。

匿名内部类可以执行正常助手类做不到的事情：访问`keyLabel`实例变量。该变量属于`KeyViewer`类。内部类可以访问所属类的方法和变量。

在NetBeans中创建一个名为`NewKeyViewer`的Java新文件，将`com.java24hours`设置为该程序的包。输入程序清单20.2中的源代码，然后保存。

程序清单20.2 `NewKeyViewer.java`的源代码

```
1: package com.java24hours;  
2:  
3: import javax.swing.*;
```

```

4: import java.awt.event.*;
5: import java.awt.*;
6:
7: public class NewKeyViewer extends JFrame {
8:     JTextField keyText = new JTextField(80);
9:     JLabel keyLabel = new JLabel("Press any key in the text
field.");
10:
11:     public NewKeyViewer() {
12:         super("NewKeyViewer");
13:         setLookAndFeel();
14:         setSize(350, 100);
15:         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16:         keyText.addKeyListener(new KeyAdapter() {
17:             public void keyTyped(KeyEvent input) {
18:                 char key = input.getKeyChar();
19:                 keyLabel.setText("You pressed " + key);
20:             }
21:         });
22:         BorderLayout bord = new BorderLayout();
23:         setLayout(bord);
24:         add(keyLabel, BorderLayout.NORTH);
25:         add(keyText, BorderLayout.CENTER);
26:         setVisible(true);
27:     }
28:
29:     private void setLookAndFeel() {
30:         try {
31:             UIManager.setLookAndFeel(
32: "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
33:         );
34:         } catch (Exception exc) {
35:             // ignore error
36:         }
37:     }
38:
39:     public static void main(String[] arguments) {
40:         new NewKeyViewer();
41:     }
42: }

```

代码第16~21行创建并使用了匿名内部类。它使用KeyListener接口中唯一的方法来监控键盘输入，并通过更新keyLabel实例变量来显示

该输入。

该程序的输出如图20.2所示。

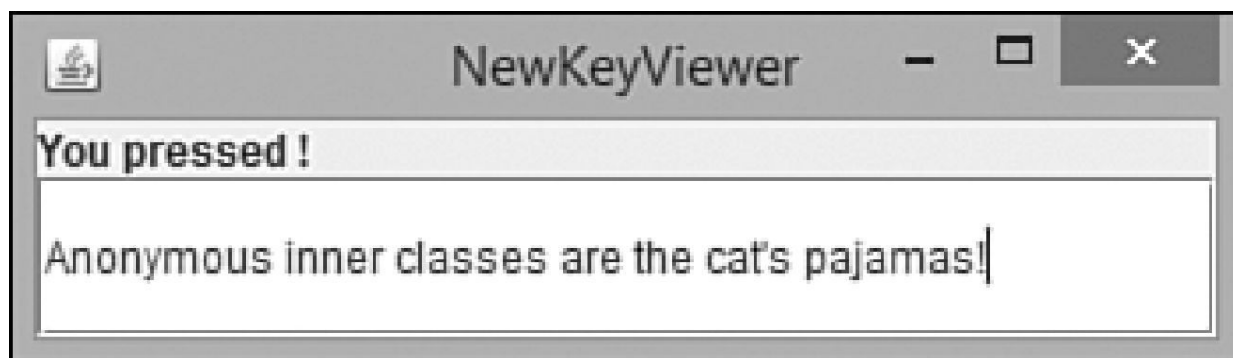


图20.2 使用NewKey Viewer应用程序监控键盘输入

匿名内部类不能定义构造函数，因此相较于非匿名的内部类，其限制更多。

匿名内部类是Java语言中的一个复杂特性，在查看程序的源代码时会很难理解，但是它可以让程序更为简洁。有经验的Java程序员会经常使用匿名内部类。

20.2 闭包

新发布的Java 8添加了近年来呼声最高的一个特性：闭包。

闭包也成为lambda表达式，在其他条件满足的情况下，它是可以只使用一个->操作符创建带有单个参数的对象。

下面是一个例子：

```
Runnable runner = () ->{ System.out.println("Run!"); };
```

这行代码创建了一个实现了**Runnable**接口的对象，而且**run()**方法等同于下面的代码：

```
void run() {  
    System.out.println("Run!");  
}
```

在**lambda**表达式中，箭头操作符（->）右边的语句成为实现接口的方法。

只有当接口有一个要实现的方法时，才能这样做，比如只包含**run()**的**Runnable**。在Java中，拥有一个方法的接口现在称为函数式接口。

lambda表达式在箭头操作符的左边也有一些内容。在第一个例子中，它是一个内容为空的括号，用来引用发送给函数式接口的方法的参数。由于在**Runnable**接口中的**run()**不接收参数，因此在该表达式中不需要参数。

下面这个**lambda**表达式的例子在括号中添加了一些内容：

```
ActionListener al = (ActionEvent act) -> {  
    System.out.println(act.getSource());  
};
```

```
}
```

在一个实现了`java.awt.event`包中`ActionListener`接口的对象中，上述代码等同于下面的代码：

```
public void actionPerformed(ActionEvent act) {  
    System.out.println(act.getSource());  
}
```

`ActionListener`接口接收动作事件，比如用户点击按钮的事件。在函数式接口中的唯一方法是`actionPerformed(ActionEvent)`。这个参数是一个`ActionEvent`对象，用来表述触发事件的用户动作。

`lambda`表达式的右半部分将`actionPerformed()`方法定义为一条语句，可以显示触发事件的用户接口组件相关的信息。

表达式的左半部分将`ActionEvent`对象`act`声明为方法的参数。

对象`act`在方法的内部使用，而表达式左半部分对`act`对象的引用似乎出现在方法的作用域之外。这是因为闭包允许变量被外部方法所引用。

可以看到，`lambda`表达式的一个效果是缩短了代码。一个表达式就可以创建一个对象并实现一个接口。

借助于Java语言对目标类型（target typing）的支持，Java的这一新特性可以让代码更为简洁。

在lambda表达式中，可以推断出发送给方法的参数的类。来考虑最后一个例子。由于ActionListener函数式接口有一个方法，该方法接收一个ActionEvent对象作为其唯一的参数，因此该对象的类名可以忽略。

下面是lambada表达式的一个修改版本，它考虑到了上述情况：

```
ActionListener al = (act) -> {  
    System.out.println(act.getSource());  
}
```

接下来要创建的两个程序可以更为具体地演示在Java中引入闭包后所带来的区别。

程序清单20.3中ColorFrame程序是一个Swing图形程序，显示用来更改框架颜色的3个按钮。该程序使用了一个匿名内部类（而非lambda表达式）来监控用户在3个按钮上的单击行为。

在NetBeans中创建一个名为ColorFrame的新程序，并将com.java24hours作为包，然后输入程序清单20.3中的所有代码。

程序清单20.3 ColorFrame.java的完整源代码

```
1: package com.java24hours;
```

```

2:
3: import java.awt.*;
4: import java.awt.event.*;
5: import javax.swing.*;
6:
7: public class ColorFrame extends JFrame {
8:     JButton red, green, blue;
9:
10:    public ColorFrame() {
11:        super("ColorFrame");
12:        setLookAndFeel();
13:        setSize(322, 122);
14:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15:        FlowLayout flo = new FlowLayout();
16:        setLayout(flo);
17:        red = new JButton("Red");
18:        add(red);
19:        green = new JButton("Green");
20:        add(green);
21:        blue = new JButton("Blue");
22:        add(blue);
23:        // begin anonymous inner class
24:        ActionListener act = new ActionListener() {
25:            public void actionPerformed(ActionEvent event) {
26:                if (event.getSource() == red) {
27:                    getContentPane().setBackground(Color.RED);
28:                }
29:                if (event.getSource() == green) {
30:                    getContentPane().setBackground(Color.GREEN);
31:                }
32:                if (event.getSource() == blue) {
33:                    getContentPane().setBackground(Color.BLUE);
34:                }
35:            }
36:        };
37:        // end anonymous inner class
38:        red.addActionListener(act);
39:        green.addActionListener(act);
40:        blue.addActionListener(act);
41:        setVisible(true);
42:    }
43:
44:    private void setLookAndFeel() {
45:        try {
46:            UIManager.setLookAndFeel(
47:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
48:            );
49:        } catch (Exception exc) {
50:            // ignore error
51:        }

```

```
52:    }  
53:  
54:    public static void main(String[] arguments) {  
55:        new ColorFrame();  
56:    }  
57: }
```

该程序的运行结果如图20.3所示。

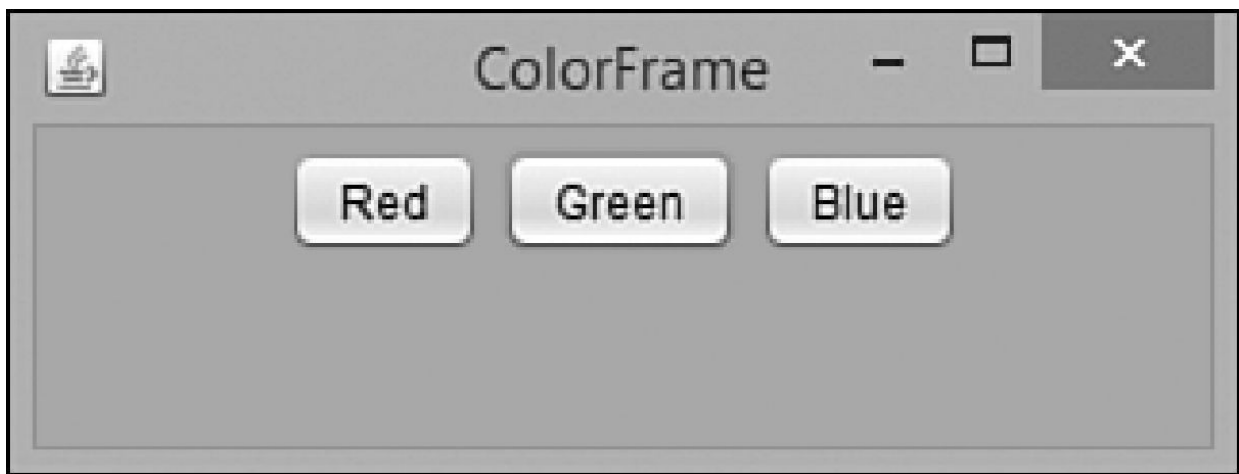


图20.3 使用匿名内部类来监控动作事件

代码的第24~26行使用一个匿名内部类为ColorFrame类创建了一个事件监听器。这是一个没有名字的对象，并实现了ActionListener接口的唯一方法：actionPerformed(ActionEvent)。

在该方法中，通过调用框架的getConentPane()方法来检索其内容面板。匿名内部类可以访问其所属类中的方法和实例变量。而单独的辅助类则无法做到这一点。

内容面板的setBackground(color)方法将框架的背景色修改为参数指定的颜色。3个按钮的颜色不发生变化。

现在看一下NewColorFrame应用程序。在NetBeans创建一个名为NewColorFrame的新程序，然后输入程序清单20.4中的所有文本。

程序清单20.4 NewColorFrame.java的完整源代码

```
1: package com.java24hours;
2:
3: import java.awt.*;
4: import java.awt.event.*;
5: import javax.swing.*;
6:
7: public class NewColorFrame extends JFrame {
8:     JButton red, green, blue;
9:
10:    public NewColorFrame() {
11:        super("NewColorFrame");
12:        setLookAndFeel();
13:        setSize(322, 122);
14:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15:        FlowLayout flo = new FlowLayout();
16:        setLayout(flo);
17:        red = new JButton("Red");
18:        add(red);
19:        green = new JButton("Green");
20:        add(green);
21:        blue = new JButton("Blue");
22:        add(blue);
23:        // begin lambda expression
24:        ActionListener act = (event) -> {
25:            if (event.getSource() == red) {
26:                getContentPane().setBackground(Color.RED);
27:            }
28:            if (event.getSource() == green) {
29:                getContentPane().setBackground(Color.GREEN);
30:            }
31:            if (event.getSource() == blue) {
32:                getContentPane().setBackground(Color.BLUE);
33:            }
34:        };
35:        // end lambda expression
36:        red.addActionListener(act);
```

```
37:         green.addActionListener(act);
38:         blue.addActionListener(act);
39:         setVisible(true);
40:     }
41:
42:     private void setLookAndFeel() {
43:         try {
44:             UIManager.setLookAndFeel(
45: "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
46:         );
47:         } catch (Exception exc) {
48:             // ignore error
49:         }
50:     }
51:
52:     public static void main(String[] arguments) {
53:         new NewColorFrame();
54:     }
55: }
```

`NewColorFrame`程序在第24~34行实现了一个动作监听器。你无需知道`ActionListener`接口的方法名称就可以在程序中使用，而且也无需指定`ActionEvent`的类。

`lambda`表达式支持函数式编程。函数式编程是软件设计中的一种方法，直到现在才开始被Java程序员使用。

本节介绍了`lambda`表达式的基本语法，以及在程序中使用它的两种常见方法。

但是函数式编程在Java中是一个功能强大和革命性的主题，以至于它也是本书的主题。

此时，你应该能够识别出**lambda**表达式，并能够使用它们来实现只有单个方法的接口（也成为函数式接口）。

20.3 总结

内部类、匿名内部类和**lambda**表达式是Java语言中最难学习的地方。具有Java编程经验的程序员来说，这些内容相当具有吸引力，因为他们可以使用这些强大的特性编写出代码行更少、功能更强的程序。

当你能够自行编写**lambda**表达式时，就能够从内部类和匿名内部类的使用中获益。

非匿名内部类的结构与单独的辅助类很像，但是它是放在另一个类的里面，与实例变量、类变量、实例方法和类方法一起存放。与辅助类不同，它可以访问其所属类的私有变量和方法。

借助于匿名内部类，我们无需再在类中放置只用一次的对象。这样的例子是在Swing中关联到用户界面组件的事件监听器。

lambda表达式的外观看起来很简洁，只需要一个箭头操作符（->）就能创建。但是它们可以让程序员做大量的事情。多年以来，具有其他语言编程经验的程序员就一直呼吁Java添加该功能。

20.4 问与答

问：匿名内部类很让人摸不着头脑。我必须在程序中使用它么？

答：不是。与Java语言中其他复杂的特性一样，如果你可以通过其他方式完成工作，就没有必要使用它。

但是无论如何，你都应该学习它，因为你很有可能会在Java代码中遇到它。

当你阅读由经验丰富的程序员编写的Java程序时，你经常会发现内部类和匿名内部类。因此，即使你不打算自行创建它们，也有必要了解它们是什么，以及是如何工作的。

20.5 测验

为了查看你是否掌握了本章介绍的知识，请回答下述关于内部类、匿名内部类和lambda表达式相关的问题。

20.5.1 问题

1. 是什么原因导致有些内部类是匿名的?
 - a. 它们实现了一个方法。
 - b. 它们没有名字。
 - c. 两者皆是。
2. 如果一个接口只包含一个方法，则该接口的另一个名字是什么?
 - a. 抽象接口。

b. 类。

c. 函数式接口。

3. 当lambda表达式猜测方法所使用的参数的类型时，这称之为什么？

a. 目标类型。

b. 类型转换。

c. 类推理。

20.5.2 答案

1. **c.** 内部匿名函数使用new关键字来实现接口，而没有必要再创建和命名实现接口的类。

2. **c.** 在Java 8中称为函数式接口。在早期的版本中，称之为单一抽象方法接口。

3. **a.** 目标类型可以推测任何函数式接口的方法参数的类型。

20.6 练习

要结束本章关于类的学习，请完成下列练习：

重写第16章的LottoEvent程序，为事件监听器使用lambda表达式。

编写一个**Swing**程序，当次单击按钮时抛出一个**6**面骰子。为事件监听器使用**lambda**表达式，然后为骰子使用**Dice**内部类。

有关为完成这些练习而编写的**Java**程序，请访问本书的配套网站 www.java24hours.com。

第21章 读写文件

本章介绍如下内容：

- 将文件中的字节读取到程序中；
- 在计算机上创建新文件；
- 将字节数组存储到文件中；
- 更改存储在文件中的数据。

在计算机中有很多表示数据的方式，读者已经通过创建对象使用过其中一种。对象包含以变量方式存储的数据以及对象的引用，还包含使用数据来完成任务的方法。

要使用其他类型的数据，如存储在硬盘中的文件和Web服务器中的文档，可使用java.io包中的类。其中的io表示input/output，这些类用于访问数据源，如硬盘、DVD或计算机内存。

可以使用称为流（将信息从一个地方传到另一个地方的对象）的通信系统，将数据传入程序或从程序传出数据。

21.1 流

要在Java程序中永久地保存数据或要检索已存储的数据，至少必须有一个流。

流是一种对象，将信息从一个地方发送到另一个地方。其名称来自溪流，溪流可以将鱼、船和工业污染物都从一个地方带到另一个地方。

流可以连接到各种数据源，包括计算机程序、硬盘、Internet服务器、计算机内存和优盘。在学会如何使用流处理一种数据后，就能够以相同的方式处理其他类型的数据。

本章将使用流来读写计算机上文件中的数据。

有如下两种类型的流。

- 输入流：从数据源读取数据。
- 输出流：将数据写入数据源。

所有输入和输出流都由字节组成，字节是0~255的整数。可以用这种格式来表示数据，如可执行程序、字处理文档和MP3音乐文件，但这只是字节可表示的一小部分数据。字节流用于读写这种数据。

By the Way

注意

Java类文件以字节码（bytecode）的形式存储为字节。Java解释器运行字节码，而字节码并不一定是由Java语言生成。Java虚拟机也能运行其他语言生成的字节码，包括NetRexx和Jython语言。你可能听说过Java虚拟机也被称为字节码解释器。

一种更专用的数据表示方式是字符——字母、数字、标点符号等。读写文本数据源时，可以使用字符流。

无论是使用字节流、字符流还是其他类型的信息，整体过程是相同的：

- 创建一个与数据相关联的流对象；
- 调用流的方法，将信息加入流中或从流中取出信息；
- 调用流对象的`close()`方法关闭流。

21.1.1 文件

在Java中，文件用**File**类表示，它也位于**java.io**包中。可以读取硬盘、CD-ROM或其他存储设备中的文件。

File对象可以表示已有的文件或要创建的文件。要创建**File**对象，使用文件名作为构造函数的参数，如下例所示：

```
File bookName = new File("address.dat");
```

这条语句创建一个位于当前文件夹中的**address.dat**文件的对象。也可以在文件名中包括路径：

```
File bookName = new File("data\\address.dat");
```

By the Way

注意

上述代码适用于Windows系统，它使用反斜线（\）字符作为文件名和路径的分隔符。Linux和其他基于UNIX的系统使用斜线（/）字符作为分隔符。编写Java程序时，要以适用于任何操作系统的方式引用文件，可使用类变量File.pathSeparator而不是反斜线和斜线，如下面的语句所示：

```
File bookName = new File("data" + File.pathSeparator
    + "address.dat");
```

有了File对象后，就可以调用该对象的几个有用的方法。

- exists(): 如果文件存在，返回true；否则返回false。
- getName(): 将文件名作为String返回。
- length(): 将文件长度作为long值返回。
- createNewFile(): 如果文件不存在，创建它。
- delete(): 如果文件存在，将其删除。
- renameTo(File): 使用通过参数指定的File对象的名称来重命名文件。

也可以使用**File**对象表示系统中的文件夹而不是文件。为此，在构造函数**File**中指定文件夹名，这可以是绝对路径（如 `C:\MyDocuments\`），也可以是相对路径（如 `java\database`）。

有了代表文件夹的对象后，就可以调用其**listFiles()**方法来看文件夹的内容。该方法返回一个**File**对象数组，表示文件夹包含的每个文件和子文件夹。

21.1.2 从流中读取数据

本章的第一个项目将使用输入流从文件中读取数据。为此，可使用**FileInputStream**类，它代表可通过它从文件中读取字节的输入流。

要创建文件输入流，可调用**FileInputStream()**构造函数并将一个文件名或**File**对象作为参数。

在访问文件时如果发生了错误，将引发**IOException**异常，从而导致读取或写入文件的方法失败。与读写文件相关的许多方法都会生成这个异常，因此通常会用到**try-catch**语句块。

流是Java中的一种资源，因此在不再使用时，必须将其关闭。当程序运行时，让流处于打开状态对Java虚拟机来讲是一种巨大的资源损失。

可以使用一种称为**try-with-resource**的特殊**try**语句确保资源（比如文件输入流）在不再使用时能够关闭。**try**语句的后面是括号。在括号里面的是一条或多条Java语句，它声明了通过资源读写数据的变量。

下面这个例子使用**stream**文件输入流读取**cookie.web**文本文件：

```
File cookie = new File("cookie.web");
try (FileInputStream stream = new FileInputStream(cookie)) {
    System.out.println("Length of file: " + cookie.length());
} catch (IOException ioe) {
    System.out.println("Could not read file.");
}
```

因为流位于try语句中，当try-catch语句执行完毕后，流将自动关闭（如果还没有关闭的话）。

文件输入流以字节方式读取数据。要读取单个字节，可调用流的read()方法且不指定任何参数。如果因已到达文件末尾，导致流中没有可读取的字节，将返回字节值-1。

读取输入流时，从流中的第一个字节开始读取，如文件的第一个字节。可以调用带有一个参数的skip()方法来跳过一些字节，该参数是一个int参数，表示要跳过的字节数。下面的语句跳过scanData流中接下来的1024个字节：

```
scanData.skip(1024);
```

如果要一次读取多个字节，可以这样做：

- 创建一个字节数组，其大小等于要读取的字节数；
- 使用该数组作为参数调用read()方法，将使用从流中读取的字节填充该数组。

下面创建一个从MP3音频文件读取ID3数据的应用程序。MP3是一种非常流行的音乐文件格式，因此经常在ID3文件的末尾添加128个字节，用于存储与歌曲相关的信息，如歌名、艺术家及其所属的唱片。

应用程序ID3Reader使用文件输入流来读取MP3文件，它跳过最后128个字节前的所有内容。然后检查余下的字节是否包含ID3数据，如果包含，前3个字节应为84、65和71。

***By the
Way***

注意

Java支持的Unicode标准字符集包含ASCII字符集，在ASCII字符集中，这3个数字分别代表大写字母T、A和G。

创建一个名为ID3Reader的Java空文件，然后输入程序清单21.1中的所有文本。

程序清单21.1 ID3Reader.java的源代码

```
1: package com.java24hours;
2:
3: import java.io.*;
4:
5: public class ID3Reader {
6:     public static void main(String[] arguments) {
7:         File song = new File(arguments[0]);
8:         try (FileInputStream file = new FileInputStream(song))
9:         {
10:             int size = (int) song.length();
11:             file.skip(size - 128);
12:             byte[] last128 = new byte[128];
13:             file.read(last128);
14:             String id3 = new String(last128);
```

```

14:         String tag = id3.substring(0, 3);
15:         if (tag.equals("TAG")) {
16:             System.out.println("Title: " +
id3.substring(3, 32));
17:             System.out.println("Artist: " +
id3.substring(33, 62));
18:             System.out.println("Album: " +
id3.substring(63, 91));
19:             System.out.println("Year: " +
id3.substring(93, 97));
20:         } else {
21:             System.out.println(arguments[0] + " does not
contain"
22:                                 + " ID3 info.");
23:         }
24:         file.close();
25:     } catch (IOException ioe) {
26:         System.out.println("Error -- " + ioe.toString());
27:     }
28: }
29: }

```

在作为应用程序运行该类之前，必须将一个**MP3**文件指定为命令行参数。该程序可以对任何**MP3**文件都生效，比如**Come on and Getit.mp3**（这是由**Marion Black**演唱的一首歌曲）。如果你的系统中有这首歌曲（应该有），则图21.1就是**ID3Reader**应用程序运行时的输出。

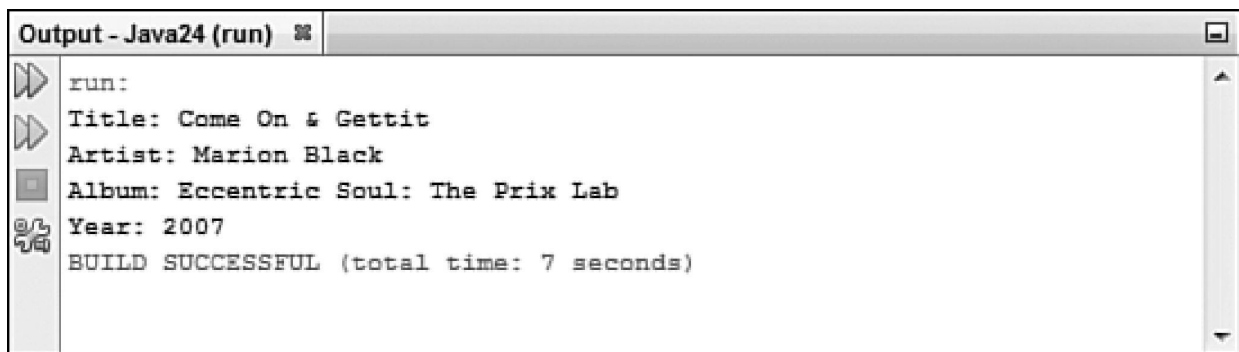


图21.1 运行ID3Reader应用程序

Did you Know?

提示

如果你的计算机中没有Come on and Getit.mp3文件（在我看来，这是严重的错误），可在Creative Commons查找MP3歌曲，Creative Commons可通过<http://search.creativecommons.org>搜索到。

Creative Commons是一组版权许可协议，规定了可如何分发、编辑或复制歌曲和书籍等作品。Rock Proper站点（www.rockproper.com）提供了大量MP3专辑，这些专辑可以在Creative Commons的许可下共享。

在程序清单21.1的第11~12行，应用程序读取MP3文件的最后128个字节，并将其存储到一个字节数组中。该数组在第13行被用于创建一个String对象，后者包含这些字节表示的字符。

如果字符串中的前3个字符为“TAG”，表明该MP3文件包含的ID3信息是应用程序能够识别的格式。

在第16~19行，调用字符串的substring()方法来显示字符串的各个部分。要显示的字符为ID3格式，这种格式总是将艺术家、歌曲、曲名和年度等信息存储在MP3文件最后128字节的相同位置。

有些MP3文件不包含ID3信息或包含的ID3信息为应用程序不能读取的格式。

如果Come on and Getit.mp3文件来自你购买的Eccentric Soul CD，它将包含可读取的ID3信息，因为使用音频CD创建MP3文件的程序从音乐行业数据库CDDb中读取歌曲信息。

从MP3文件的输入流中读取与ID3相关的信息后，第24行关闭这个流。使用完流后，应关闭它，这样可以节省Java解释器的资源。

By the Way

注意

读者可能想从BitTorrent服务（这是一种非常流行的文件共享服务）中查找Come on and Getit.mp3文件的拷贝，就这首MP3而言，我非常理解这种做法。然而，根据美国唱片行业协会的规定，任何人下载不属于自己的CD的MP3文件，将被起诉。因此，最好先从Amazon.com、eBay、Applet iTunes以及其他主流零售商那里购买Eccentric Soul CD。

21.1.3 缓冲输入流

对于读取输入流的程序，提高其性能的一个方法是将输入放到缓冲区中。缓冲（buffering）是将数据放到内存中供程序需要时使用的一个过程。当Java程序需要缓冲输入流中的数据时，将首先在缓冲区中查找，这比从文件等数据源读取数据快。

要使用缓冲输入流，需要创建一个输入流，如FileInputStream对象，然后使用该对象创建缓冲的流。为此，将输入流作为唯一的参数调用BufferedInputStream（InputStream）构造函数，这样，从输入流中读取数据时，数据将被存储到缓冲区中。

要从缓冲的流中读取数据，可调用其read()方法且不指定任何参数。这将返回一个0~255的整数，表示该流中下一字节的数据。如果没有字节可读取，将返回-1。

为了演示缓冲的流，将要创建的下一个程序将使用Java的一项功能：控制台输入，这项功能在很多其他语言中是没有的。

控制台输入指的是程序运行时从控制台（也叫命令行）读取字符。

System类包含一个out变量，语句System.out.print()和System.out.println()使用了该变量；它还有一个名为in的类变量，表示一个InputStream对象，该对象从键盘接收输入并以流的方式提供它们。

可以像使用其他输入流那样使用这个输入流，下面的语句创建一个缓冲的输入流，它与输入流System.in相关联：

```
BufferedInputStream bin = new BufferedInputStream(System.in);
```

接下来的项目（Console类）包含一个类方法，可在任何Java程序中用于接收控制台输入。在名为Console的Java空文件中输入程序清单21.2中的所有文本。

程序清单21.2 Console.java的完整源代码

```
1: package com.java24hours;
2:
3: import java.io.*;
4:
5: public class Console {
6:     public static String readLine() {
7:         StringBuilder response = new StringBuilder();
8:         try {
9:             BufferedInputStream bin = new
```

```

10:         BufferedInputStream(System.in);
11:         int in = 0;
12:         char inChar;
13:         do {
14:             in = bin.read();
15:             inChar = (char) in;
16:             if (in != -1) {
17:                 response.append(inChar);
18:             }
19:         } while ((in != -1) & (inChar != '\n'));
20:         bin.close();
21:         return response.toString();
22:     } catch (IOException e) {
23:         System.out.println("Exception: " +
e.getMessage());
24:         return null;
25:     }
26: }
27:
28: public static void main(String[] arguments) {
29:     System.out.print("You are standing at the end of the
road ");
30:     System.out.print("before a small brick building.
Around you ");
31:     System.out.print("is a forest. A small stream flows
out of ");
32:     System.out.println("the building and down a
gully.\n");
33:     System.out.print("> ");
34:     String input = Console.readLine();
35:     System.out.println("That's not a verb I recognize.");
36: }
37: }

```

Console类包含一个**main()**方法，用于演示如何使用它。当运行该程序时，输出结果如图21.2所示。

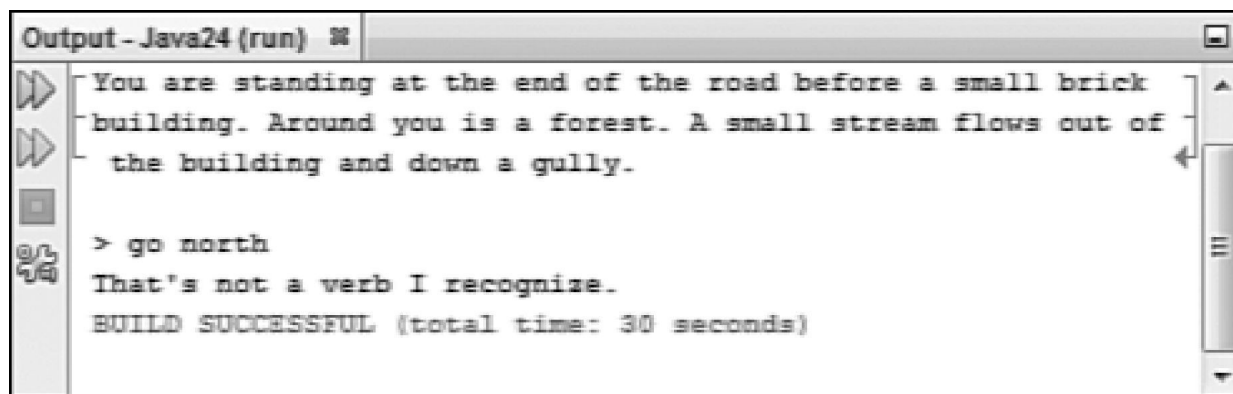


图21.2 运行Console应用程序

Console类包含一个类方法：`readLine()`，它从控制台接收字符。用户按回车键后，`readLine()`方法返回一个String对象，其中包含收到的所有字符。

By the Way

注意

Console类也是世界上最不能让人省心的一款文字冒险游戏，在该游戏中，你不能进入大楼，无法在溪流中涉水前进，也无法四处闲逛。该游戏也称为Adventure，有关该游戏的完整版本，请访问Web-Adventures，网址为www.web-adventures.org。

21.2 将数据写入流中

在java.io包中，与流相关的类都是成对出现的。对于字节流，有FileInputStream类和FileOutputStream类；对于字符流，有FileReader类和FileWriter类，还有很多其他成对的用于处理流数据的类。

要将数据写入字节流中，首先需要创建一个与输出流相关联的File对象。该文件不必是系统中现有的文件。

可以通过两种方式创建FileOutputStream。如果要在现有的文件中追加字节，使用两个参数调用构造函数FileOutputStream()：一个是代表文件的File对象，另一个是布尔值true。这样，写入流中的字节就会追加到文件的末尾。

如果要将字节写入一个新文件中，只使用一个File对象作为参数调用构造函数FileOutputStream()。

有了输出流后，就可以调用不同的write()方法来写入字节。

- 用一个字节作为参数调用write()方法时，将该字节写入流中。
- 用一个字节数组作为参数调用write()方法时，将数组的所有字节写入流中。
- 给write(byte[], int, int)方法指定3个参数：一个字节数组、一个表示要写入流中的数组的第一个元素的整数、要写入的字节总数。

下面的语句创建一个包含10个字节的字节数组，并将最后5个字节写入到输出流：

```
File dat = new File("data.dat");
FileOutputStream datStream = new FileOutputStream(dat);
byte[] data = new byte[] { 5, 12, 4, 13, 3, 15, 2, 17, 1, 18 };
datStream.write(data, 5, 5);
```

将数据写入到流中时，可以调用**String**对象的**getBytes()**方法，将文本转换为字节数组，如下面的示例所示：

```
String name = "Puddin N. Tane";  
byte[] nameBytes = name.getBytes();
```

在将字节写入到流中后，可以调用流的**close()**方法将其关闭。

现在编写一个简单的应用程序**ConfigWriter**，它通过将字节写入到文件输出流中的方式，将几行文本存储到一个文件中。创建一个名为**ConfigWriter**的Java空文件，然后输入程序清单21.3中的所有文本。

程序清单21.3 ConfigWriter.java的完整源代码

```
1: package com.java24hours;  
2:  
3: import java.io.*;  
4:  
5: public class ConfigWriter {  
6:     String newline = System.getProperty("line.separator");  
7:  
8:     public ConfigWriter() {  
9:         try {  
10:             File file = new File("program.properties");  
11:             FileOutputStream fileStream = new  
FileOutputStream(file);  
12:             write(fileStream, "username=max");  
13:             write(fileStream, "score=12550");  
14:             write(fileStream, "level=5");  
15:             fileStream.close();  
16:         } catch (IOException ioe) {  
17:             System.out.println("Could not write file");  
18:         }  
19:     }  
20:  
21:     void write(FileOutputStream stream, String output)
```

```
22:         throws IOException {
23:
24:         output = output + newline;
25:         byte[] data = output.getBytes();
26:         stream.write(data, 0, data.length);
27:     }
28:
29:     public static void main(String[] arguments) {
30:         ConfigWriter cw = new ConfigWriter();
31:     }
32: }
```

运行该应用程序时，将创建一个名为`program.properties`的文件，该文件包含下面3行文本：

```
Output ▼
username=max
score=12550
level=5
```

该文件是在第10行创建的，它与第11行中的一个文件输出流相关联。这3个属性是在第12~14行写入到流中的。

在没有指定其他文件夹的情况下，在NetBeans中运行的应用程序会将它创建的文件保存到项目的主文件夹中。要在NetBeans中查看`program.properties`文件，可以在Projects面板中单击File标签。该文件位于顶级的Java24文件夹中，如图21.3所示。

双击文件名，可以在NetBeans源代码编辑器中打开它。

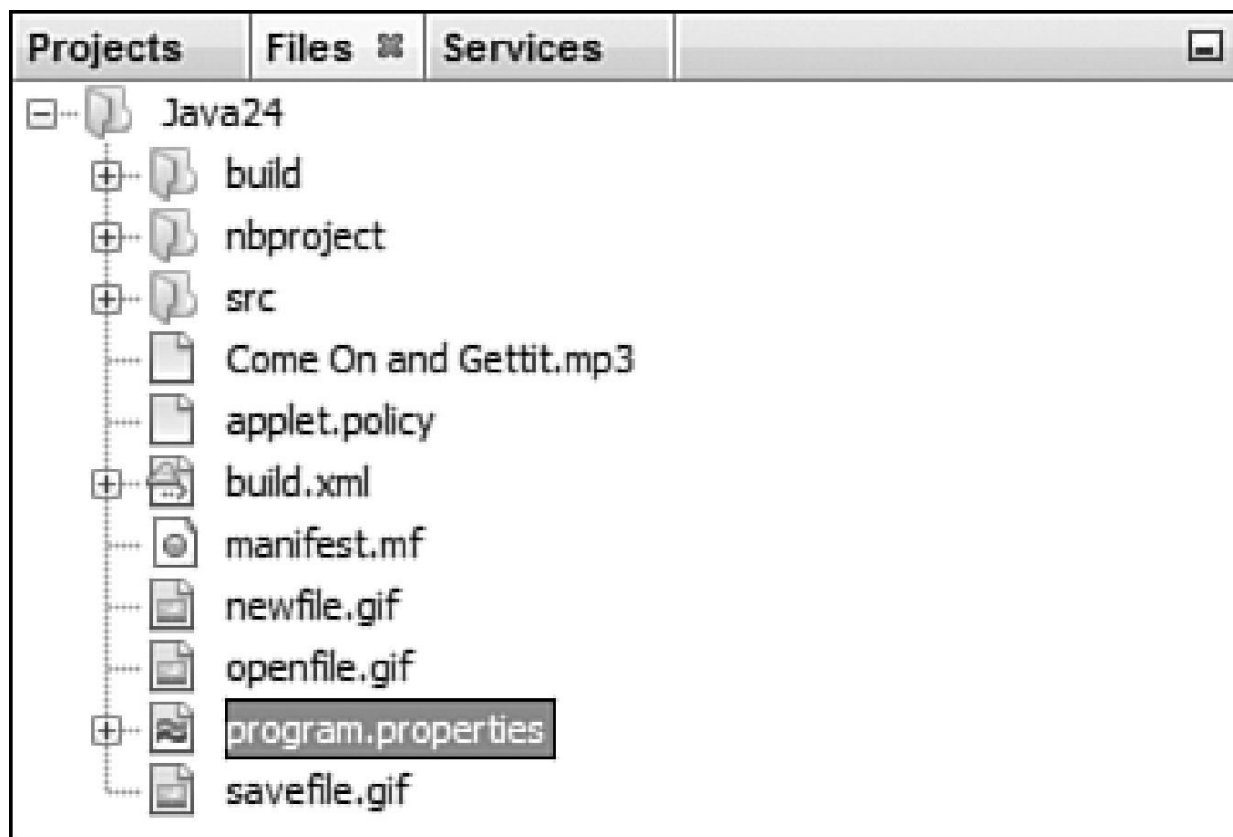


图21.3 查找program.properties文件

21.3 读写配置属性

当使用命令行参数对Java程序进行配置之后（就像前面几章中创建的应用程序那样），Java程序将更有用。java.util包中有一个Properties类，可用于从另外一个源（文本文件）中载入配置设置。

在Java中，可以像其他文件那样读取属性文件：

- 创建一个代表该文件的File对象；
- 使用该File对象创建一个FileInputStream对象；
- 调用load()方法从输入流中检索属性。

属性文件由一组属性名、等号和属性值组成，如下例所示：

```
username=lepton  
lastCommand=open database  
windowSize=32
```

每个属性占一行，因此上述内容将属性username、lastCommand和windowSize的值分别设置为lepton、open database和32（ConfigWriter应用程序也使用相同的格式）。

下面的代码载入一个名为config.dat的属性文件：

```
File configFile = new File("config.dat");  
FileInputStream inStream = new FileInputStream(configFile);  
Properties config = new Properties();  
config.load(inStream);
```

配置设置也称为属性，作为字符串存储在Properties对象中。每个属性都用一个键来标识。方法getProperty()根据键来检索属性，如下面的语句所示：

```
String username = config.getProperty("username");
```


由于属性被存储为字符串，要将其用作数字值，必须采用某种方式对其进行转换，如下面的代码所示：

```
String windowProp = config.getProperty("windowSize");
int windowSize = 24;
try {
    windowSize = Integer.parseInt(windowProp);
} catch (NumberFormatException exception) {
    // do nothing
}
```

要存储属性，可调用`setProperty()`方法并指定两个参数——键和值：

```
config.setProperty("username", "max");
```

可以调用`Properties`对象的`list(PrintStream)`方法来显示所有的属性。`PrintStream`是`System`类中`out`类的变量。在本书前面，我们已经在`System.out.println()`语句中使用`out`来显示输出。下面的代码将调用`list()`方法来显示所有的属性。

```
config.list(System.out);
```

在对属性作出修改之后，可将其存回到文件中：

- 创建一个代表文件的File对象;
- 根据该File对象创建一个FileOutputStream对象;
- 调用方法store(OutputStream, String)将属性存储到指定的输出流中, 而且参数String是属性文件的描述。

接下来创建ConfigWriter应用程序, 它将多个文件程序设置写入到一个文件中。而Configurator应用程序将这些属性设置读入到一个Java属性文件中, 并添加一个名为runtime的新属性 (具有当前日期和时间), 然后保存该文件。

创建一个名为Configurator的Java空文件, 然后输入程序清单21.4中的所有文本。

程序清单21.4 Configurator.java的完整源代码

```
1: package com.java24hours;
2:
3: import java.io.*;
4: import java.util.*;
5:
6: public class Configurator {
7:
8:     public Configurator() {
9:         try {
10:             // load the properties file
11:             File configFile = new File("program.properties");
12:             FileInputStream inStream = new
FileInputStream(configFile);
13:             Properties config = new Properties();
14:             config.load(inStream);
15:             // create a new property
16:             Date current = new Date();
17:             config.setProperty("runtime", current.toString());
18:             // save the properties file
19:             FileOutputStream outStream = new
                ➔ FileOutputStream(configFile);
20:             config.store(outStream, "Properties settings");
21:             inStream.close();
```

```

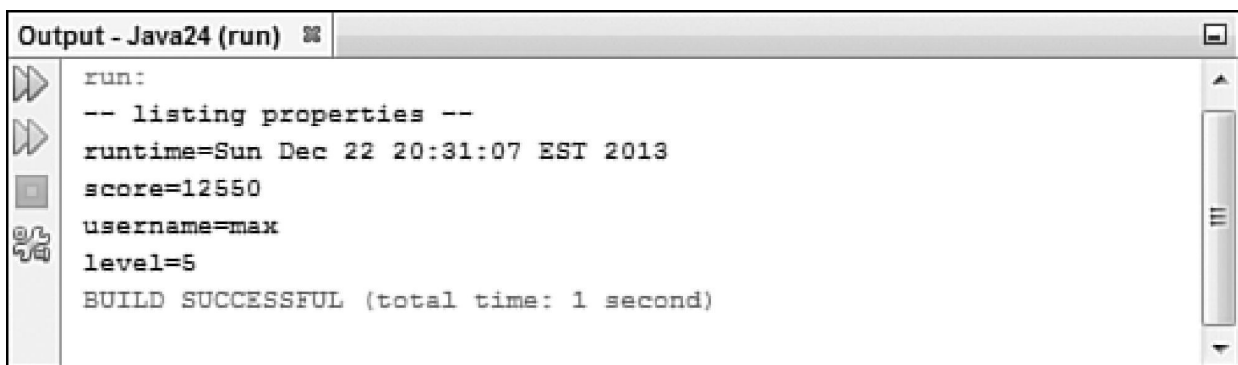
22:         config.list(System.out);
23:     } catch (IOException ioe) {
24:         System.out.println("IO error " +
ioe.getMessage());
25:     }
26: }
27:
28: public static void main(String[] arguments) {
29:     Configurator con = new Configurator();
30: }
31: }

```

在这个应用程序中，`program.properties`的File对象是在第11~12行创建并与一个文件输入流相关联的。该文件的内容是在第13~14行从流中载入到Properties对象中的。

用于当前时间和时期的一个新属性是在第16~17行创建的。第19行将该文件与一个输出流关联起来。第20行将整个属性文件写入到该流中。

运行该应用程序的输出结果如图21.4所示。



```

Output - Java24 (run)
run:
-- listing properties --
runtime=Sun Dec 22 20:31:07 EST 2013
score=12550
username=max
level=5
BUILD SUCCESSFUL (total time: 1 second)

```

图21.4 运行Configurator应用程序

文件`program.properties`现在应该包含如下文本:

```
Output ▼  
#Properties settings  
#Sun Dec 22 20:31:07 EDT 2013  
runtime=Sun Dec 22 20\:31\:07 EDT 2013  
score=12550  
level=5  
username=max
```

上面用到的反斜线字符（\）格式与应用程序的输出不同，可以确保属性文件被正确地存储。

21.4 总结

本章介绍了用于读写字节的输入流和输出流，这是通过流提供数据的最简单方式。

`java.io`包有很多用于以其他方式处理流的类；还有一个名为`java.net`的包，它可以让你通过Internet连接读写流。

字节流的用途广泛，因为可以轻松地将字节转换为其他数据类型，如整数、字符和字符串。

本章的第一个项目（应用程序ID3Reader）从流中读取字节，然后将其转换为字符串，因为以这种格式从歌曲（如Marian Black在专辑EccentricSoul中演唱的Come on and Getit）中读取ID3数据更加容易。

21.5 问与答

问：为什么本章的一些字节流方法使用整型参数？不是应使用字节型参数吗？

答：流中的字节与byte类表示的字节是有区别的。在Java中，byte变量的取值范围为-127~128，但流中的字节的取值范围为0~255。因此，处理字节时，经常需要使用int类型，它可以存储128~255的值，而byte变量不能。

21.6 测验

为检验是否掌握了本章的大部分知识，请回答下面关于Java流的问题。

21.6.1 问题

1. 下面哪种方法可用于将字节数组转换为字符串？
 - a. 调用数组的toString()方法。
 - b. 将每个字节转换为字符，再将每个字符赋给String数组的元素。
 - c. 用数组作为参数调用构造函数String()。
2. 哪种流用于在Java程序中读取文件？
 - a. 输入流。
 - b. 输出流。

c. 都可以。

3. File类的哪个方法可用于确定文件的长度？

a. `getSize()`。

b. `read()`。

c. `length()`。

21.6.2 答案

1. c. 可以像答案b那样单独处理每个字节，但是可以使用其他数据类型来更简单地创建字符串。

2. a. 使用File对象创建一个输入流，或调用输入流的构造函数并提供文件名作为参数来创建输入流。

3. c. 该方法返回一个long值，表示流中的字节数。

21.7 练习

为体验跋涉到另一条河流的新鲜感，通过下面练习来检测其水质。

- 编写一个应用程序，它读取一个文件夹中所有MP3文件的ID3标记，并使用艺术家、曲名和唱片信息（如果有的话）来重命名这些文件。

- 编写一个应用程序，它读取一个Java程序的源文件，然后不加修改地写入到一个新文件中。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第22章 利用JAX-WS开发Web服务

本章介绍如下内容：

- 定义Web服务的Java接口；
- 将接口应用到Java类；
- 在Internet上部署Web服务；
- 查看Web服务契约；
- 开发Web服务客户端。

由于Internet无处不在，将数以百万计的台式计算机、Web服务器、电话、视频游戏机，以及其他设备互连起来的诉求，催生了Web服务。Web服务是一种经由HTTP（网络协议）与其他软件进行通信的软件。

实现上述连接的一个创新特性是用于XML Web服务的Java API（JAX-WS）。JAX-WS是一组Java类和Java包，它可以创建对Web服务发出请求的客户端，以及接受这些请求的服务。

JAX-WS支持使用简单对象访问协议（Simple Object Access Protocol, SOAP）和表述性状态转移（Representational State Transfer, REST）实现的Web服务。JAX-WS大大简化了支持这些协议的任务。作为一名程序员，你只需要创建Java对象，并调用方法来使用Web服务即可，其他事宜将在幕后进行处理。

22.1 定义服务端点接口

创建JAX-WS Web服务的第一步是创建服务端点接口（Service Endpoint Interface），这是一个Java接口，它定义了客户端在使用Web服务时能够调用的方法。

本章将要开发的SquareRootServer Web服务可以处理两类简单的任务：

- 计算一个数的平方根；
- 显示当前的日期和时间。

接口是一组方法，它提供了名称、参数和返回类型，但是没有包含实现方法的代码。接口充当对象之间的契约：如果一个对象实现了该接口，其他对象就会知道它们可以调用该对象中接口的所有方法。

在第15章中，我们已经在任何这样的Java类中实现了ActionListener接口，即当有按钮被单击时，需要接收操作事件的Java类。

在这个项目中，我们要处理的是契约的其他方面。SquareRootServer接口定义了square Root（double）和getTime()这两个方法，而且这两个方法必须出现在实现了Web服务的类中。

下面的语句定义了接口：

```
public interface SquareRootServer {  
    double getSquareRoot(double input);  
    String getTime();  
}
```

在接口中定义方法时，跟在方法后面的是分号而不是将语句块括起来的“{”和“}”字符。接口中没有定义方法的行为，而是由实现了该接口的类来处理。

由于这些方法作为JAX-WS Web服务被调用，因此必须在每个方法前面添加一个额外的修饰词：`@WebMethod`注解。

```
public interface SquareRootServer {  
    @WebMethod double getSquareRoot(double input);  
    @WebMethod String getTime();  
}
```

使用注解来简化Java代码

注解（annotation）是一种巧妙的注释（comment）形式，可以被Java解释器、编译器和编程工具所理解。注解可以定义不属于程序一部分的程序信息，使得当程序在编译或运行时，这些程序信息能够触发行为。

注解使用@符号打头，后面是注解的名字。

最常见的一个注解是`@Override`，它表示一个方法覆盖了超类的方法。下面是一个例子：

```
@Override
```

```
public void paintComponent(Graphics comp) {  
    // definition of method here  
}
```

如果期间有错误，则它不会覆盖方法（当使用了错误的类型、错误的方法名或者参数个数不一致时，将发生这种错误），编辑器可以捕获该错误（如果没有注解，则编译器不能检测到这类问题）。

`@WebMethod`注解表示一个方法可以作为Web服务调用。`SquareRootServer`接口也使用了一个`@WebService`注解，来指示接口定义了一个服务端点接口。

注解也可以接收供将来自定义使用的参数。`SquareRootServer`包含一个最终（`final`）注解：

```
@SOAPBinding(style = Style.RPC)
```

该注解在Web服务和调用该服务的客户端程序之间定义了一个契约，本章后面将会详细介绍。

现在，开始编写Web服务的代码。在NetBeans中创建一个Java空文件，其类名为`SquareRootServer`，其包名为`com.java24hours.ws`。然后在该文件中输入程序清单22.1中的内容。

程序清单22.1 `SquareRootServer.java`的完整源代码

```
1: package com.java24hours.ws;
2:
3: import javax.jws.*;
4: import javax.jws.soap.*;
5: import javax.jws.soap.SOAPBinding.*;
6:
7: @WebService
8:
9: @SOAPBinding(style = Style.RPC)
10:
11: public interface SquareRootServer {
12:     // get the square root of a number
13:     @WebMethod double getSquareRoot(double input);
14:
15:     // get the current time and date as a string
16:     @WebMethod String getTime();
17:
18: }
```

该类位于com.java24hours.ws包中，这样在部署Web服务之后，其他软件能够通过Internet轻松访问该服务。

完成接口的定义之后，开始编写实现接口两个方法（getSquareRoot()和getTime()）的代码。

22.2 创建服务实现Bean

实现了服务端点接口的Java类被称为服务实现Bean（Service Implementation Bean）。在JAX-WS的学习过程中，遇到一些新奇的术语是不可避免的。

SquareRootServerImpl类实现了SquareRootServer接口，如下所示：

```
public class SquareRootServerImpl implements SquareRootServer {
```

这表示你要创建的类必须包含接口中的所有方法，而且每一个都有适当的参数。

方法`getSquareRoot (double)`和`getTime()`是使用前面学习的技术来实现的。

类中唯一比较新鲜的地方是后面的注解，它出现在类语句之前：

```
@WebService(endpointInterface =  
"com.java24hours.ws.SquareRootServer")
```

该注解表示，类是名为`com.java24hours.ws.SquareRootSever`的服务端点接口的服务实现Bean。你必须使用完整的类名，其中包括其包的名字。

注意，注解后面跟的不是分号，这与语句不同。

开始编写该类：在`com.java24hours.ws`包中创建一个名为`SquareRootServerImpl`的Java空文件，然后输入程序清单22.2中的所有内容。

程序清单22.2 `SquareRootServerImpl.java`的完整源代码

```
1: package com.java24hours.ws;
2:
3: import java.util.*;
4: import javax.jws.*;
5:
6: @WebService(endpointInterface =
"com.java24hours.ws.SquareRootServer")
7:
8: public class SquareRootServerImpl implements SquareRootServer {
9:
10:     public double getSquareRoot(double input) {
11:         return Math.sqrt(input);
12:     }
13:
14:     public String getTime() {
15:         Date now = new Date();
16:         return now.toString();
17:     }
18: }
```

在创建了这两个类之后，即可准备启用该服务，以便其他软件来调用。

Watch Out!

警告：

服务实现Bean的名字源于JavaBeans，后者是一个特殊的Java类，在Java企业版中用作可重用的软件组件。然而，就JAX-WS而言，对Bean的引用有点不是很合适。任何Java对象，只要遵循Web服务方法和规则，而且创建时带有合适的注解，就可以作为服务实现Bean。

22.3 发布Web服务

JAX-WS Web服务可以使用诸如BEA WebLogic、GlassFish、JBoss和Jetty这样的Java应用服务器来部署。如果在支持这些服务器的开发环境中创建了SquareRootServer Web服务，此时就可以准备启用Web服务。

你也可以自己编写Java应用程序，使其载入Web服务，并通过Internet向其他客户端提供服务。

SquareRootServerPublisher应用程序只需要两个步骤就可以处理该任务：

- 载入实现了Web服务的类；
- 将该对象发布到Internet。

javax.xml.ws包中的EndPoint类有一个类方法publish(String, Object)，它可以部署Web服务。

该方法的第一个参数是可以访问Web服务的网络地址，对该项目而言，该网络地址是http://127.0.0.1:5335/service。该网络地址以主机名127.0.0.1打头，这被称为本地主机，因为该主机是你创建并运行Java程序的本地计算机。

网络地址的第2部分是本地主机的端口号，Web服务在该端口号等待连接。这里使用的端口号是5335，因为该端口号不太可能被计算机上其他Internet感知（Internet-aware）程序使用。

地址的最后一部分“/service”是路径。每一个Web服务必须有一个唯一的路径。如果在你的计算机上运行其他Web服务，则它们的路径不能

与SquareRootServer相同。

为了部署Web服务，在com.java24hours.ws包中创建一个名为SquareRootServerPublisher的Java空文件，然后输入程序清单22.3中的所有文本。

程序清单22.3 SquareRootServerPublisher.java的完整源代码

```
1: package com.java24hours.ws;
2:
3: import javax.xml.ws.*;
4:
5: public class SquareRootServerPublisher {
6:     public static void main(String[] arguments) {
7:         SquareRootServerImpl srsi = new SquareRootServerImpl();
8:         Endpoint.publish(
9:             "http://127.0.0.1:5335/service",
10:            srsi
11:        );
12:    }
13: }
```

运行该程序时，它将在计算机上的5335端口等待连接。只要程序支持基于SOAP或REST的Web服务，则无论它是使用Java语言还是其他语言编写的，都可以调用其中的Web服务的方法。只要你的Web服务位于Internet中，任何连接到Internet的软件都可以调用其方法。

22.4 使用Web服务描述语言文件

在启用该Web服务之前，你可以使用任意的Web浏览器来测试SquareRootServerPublisher应用程序的可用性。

打开浏览器，然后输入地址<http://127.0.0.1:5335/service?wsdl>。该浏览器将显示程序清单22.4中的XML文件。该文件由刚才创建的应用程序提供。

该文件是一个使用Web服务描述语言（Web Services Description Language，WSDL）编写的服务契约。WSDL是一种XML，它可以清楚地说明Web服务的运行方式，以便服务器和客户端能够充分使用它。

在创建JAX-WX服务和客户端来访问Web服务时，没有必要必须理解WSDL。当然，最好还是看一下该文件的内容，以对基于SOAP和REST的Web服务的运行方式有一个理解。

程序清单22.4 WSDL契约

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's
version
3: is JAX-WS RI 2.2.2 in JDK 7. -->
4: <!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's
version
5: is JAX-WS RI 2.2.2 in JDK 7. -->
6: <definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
7: xmlns:tns="http://ws.java24hours.com/"
8: xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9: xmlns="http://schemas.xmlsoap.org/wsdl/"
10: targetNamespace="http://ws.java24hours.com/"
11: name="SquareRootServerImplService">
12: <types></types>
13: <message name="getSquareRoot">
14: <part name="arg0" type="xsd:double"></part>
15: </message>
16: <message name="getSquareRootResponse">
17: <part name="return" type="xsd:double"></part>
18: </message>
19: <message name="getTime"></message>
20: <message name="getTimeResponse">
21: <part name="return" type="xsd:string"></part>
22: </message>
23: <portType name="SquareRootServer">
```

```
24: <operation name="getSquareRoot" parameterOrder="arg0">
25: <input message="tns:getSquareRoot"></input>
26: <output message="tns:getSquareRootResponse"></output>
27: </operation>
28: <operation name="getTime" parameterOrder="">
29: <input message="tns:getTime"></input>
30: <output message="tns:getTimeResponse"></output>
31: </operation>
32: </portType>
33: <binding name="SquareRootServerImplPortBinding"
34: type="tns:SquareRootServer">
35: <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
36: style="rpc"></soap:binding>
37: <operation name="getSquareRoot">
38: <soap:operation soapAction=""></soap:operation>
39: <input>
40: <soap:body use="literal"
41: namespace="http://ws.java24hours.com/"></soap:body>
42: </input>
43: <output>
44: <soap:body use="literal"
45: namespace="http://ws.java24hours.com/"></soap:body>
46: </output>
47: </operation>
48: <operation name="getTime">
49: <soap:operation soapAction=""></soap:operation>
50: <input>
51: <soap:body use="literal"
52: namespace="http://ws.java24hours.com/"></soap:body>
53: </input>
54: <output>
55: <soap:body use="literal"
56: namespace="http://ws.java24hours.com/"></soap:body>
57: </output>
58: </operation>
59: </binding>
60: <service name="SquareRootServerImplService">
61: <port name="SquareRootServerImplPort"
62: binding="tns:SquareRootServerImplPortBinding">
63: <soap:address location="http://127.0.0.1:5335/service">
</soap:address>
64: </port>
65: </service>
66: </definitions>
```

WSDL之所以称为服务契约，是因为它规定了访问Web服务的方式、可以与服务交换的信息，以及所传输信息的数据类型。

WSDL契约的第13~22行定义了Web服务的方法、这些方法的参数，以及作为响应返回的数据。查看这些代码行，看是否能够找到getSquareRoot()方法所在的位置。该方法接收一个double型参数，并返回一个double型值。

契约中引用的数据类型不是Java的数据类型，它们可以用于任何支持SOAP的编程语言（没有专门针对Java的Web服务）。

By the Way

注意

由于WSDL契约定义了Web服务的细节，因此可以使用它来自动完成Web服务编程的大部分过程。Java开发工具包（JDK）包含命令行工具wsimport，它可以将WSDL文件作为输入，然后编写访问Web服务的Java类。

22.5 创建Web服务客户端

本节将创建SquareRootClient应用程序，它可以调用你前面创建的Web服务的方法。当然，该服务必须处于运行状态，以便客户端与其连接。

由于像JAX-WS库这样的Web服务技术支持SOAP、REST、HTTP和XML等标准，因此不一定非要使用Java程序来连接该Web服务。

Perl、Python、Ruby和其他语言都有支持Web服务的库。

JAX-WS库在javax.xml.ws包中提供了Service类，这是一个可以创建调用Web服务的对象的工厂（factory）。

类方法Service.create(URL,QName)创建了工厂，其参数URL和QName分别来自java.net和java.xml.namespace。

URL必须是Web服务的WSDL契约的地址：

```
URL url = new URL("http://127.0.0.1:5335/service?wsdl");
```

Watch Out!

警告：

前面提到，URI不一定非要是一个可用的网络地址，尽管http://ws.java24hours.com看起来很像是一个可用的网络地址，但是它是作为唯一的标识符出现的。我拥有域名java24hours.com，并可以控制其子域名的使用方式，因此我将http://ws.java24hours.com作为URI使用。我可以保证其他Web服务提供商不会使用该标识符。

QName是一个限定名称（qualified name），这是一个与Web服务提供者关联起来的XML标识符。限定名称包含命名空间URI和本地的标识符。

命名空间URI和URL相似，但是不一定非要作为网络地址来使用。由于平方根Web服务的包名称是com.java24hours.ws，在Java中，按照惯例，该名称与Internet主机名ws.java24hours.com关联起来，用于该Web服务的命名空间URI是http://ws.java24hours.com。

该Web服务的本地标识符是服务实现Bean的名字，而且后面添加有“Service”单词。下面是一个创建限定名称的语句：

```
QName qname = new QName(  
    "http://ws.java24hours.com/",  
    "SquareRootServerImplService"  
);
```

使用URL和限定名称则可以创建Web服务客户端工厂：

```
Service service = Service.create(url, qname);
```

该工厂有getPort（Class）方法，这个方法创建了指定类的对象。为了识别作为方法参数使用的Java类，可以使用名为class的类变量。感到困惑了？当在Java语句中看到它时，就会明白了：

```
SquareRootServer srs = service.getPort(SquareRootServer.class);
```

使用SquareRootServer.class作为参数来调用getPort()方法时，将导致工厂创建一个Square RootServer对象。该对象存储在src变量中。

可以在Java的其他对象中调用SquareRootServer对象的方法：

```
System.out.println(srs.getTime());  
System.out.println(srs.getSquareRoot(625D));
```

JAX-WS库将这些方法调用作为SOAP消息打包，然后通过Internet发送到Web服务，然后进行方法调用。

当服务响应这些调用时，它将响应打包为SOAP消息，然后通过Internet发送回去，之后再被转换为Java数据类型。

创建一个名为SquareRootClient的Java文件，然后输入程序清单22.5中的所有内容。

程序清单22.5 SquareRootClient.java的完整源代码

```
1: package com.java24hours.ws;  
2:  
3: import java.net.*;  
4: import javax.xml.namespace.*;  
5: import javax.xml.ws.*;  
6:  
7: public class SquareRootClient {  
8:     public static void main(String[] arguments) throws  
Exception {  
9:         URL url = new URL("http://127.0.0.1:5335/service?  
wsdl");  
10:         QName qname = new QName(  
11:             "http://ws.java24hours.com/",  
12:             "SquareRootServiceImplService"
```

```
13:         );  
14:         Service service = Service.create(url, qname);  
15:         SquareRootServer srs =  
service.getPort(SquareRootServer.class);  
16:  
17:         System.out.println(srs.getTime());  
18:         System.out.println(srs.getSquareRoot(625D));  
19:     }  
20: }
```

运行该客户端应用程序时，如果SquareRootPublisher应用程序也已经处于运行状态，则会看到如图22.1的输出。

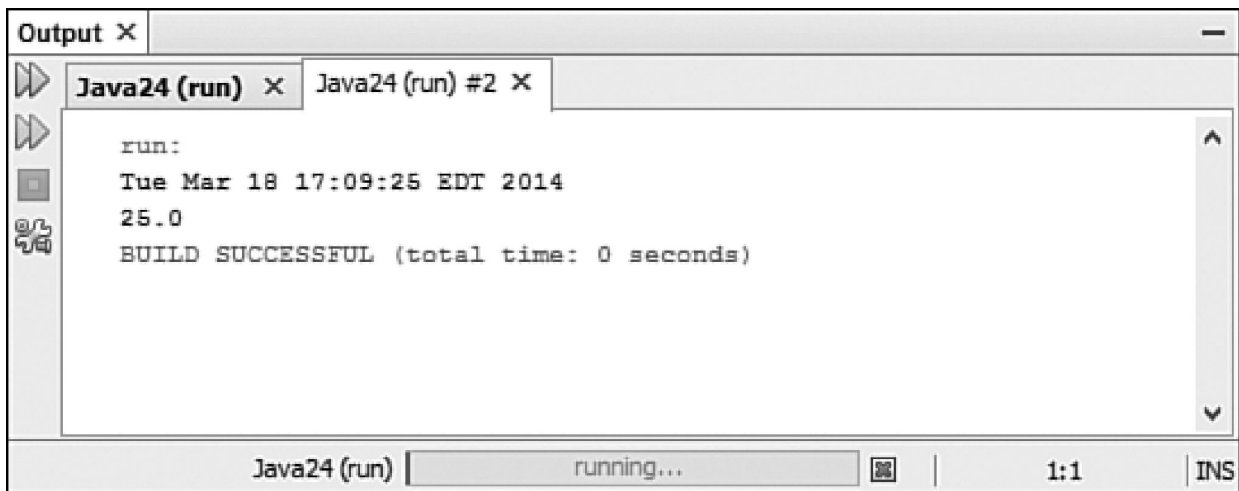


图22.1 调用Web服务并显示响应

22.6 总结

在Java中，有一类API用作基于XML的RPC（JAX-RPC）。JAX-PRC是一种允许Java对象经由Internet向另外一个对象进行远程过程调用（RPC）的技术。而JAX-WS包和类的集合则继承了该类API。

无论软件放在哪里，使用什么语言编写而成，都可以进行对其调用的能力在Web 2.0软件开发浪潮中具有重要的作用。

在20世纪90年代中期，网络开始流行之时，人们就已经享用到了Internet无处不在的连接优势，而Web 2.0则允许软件来采用这种连接优势。

本节讲解了使用JAX-WS来创建和使用Web服务的4个步骤：为服务创建一个接口（服务端点接口）、实现该服务（服务实现Bean）、在Internet上发布服务，以及创建客户端访问该服务。

许多编程工具，包括NetBeans和JDK，都可以自动创建代码，从而简化创建和访问Web服务的工作。

22.7 问与答

问：XML-RPC是如何应用于SOAP Web服务中的？

答：XML-PRC是一个协议，它可以通过HTTP来调用方法，并接收XML格式的数据。SOAP也是处理这些事情。事实上，这两个Web协议具有同一个起源。

XML-PRC诞生于一份协议的草案中，而该协议最终成为SOAP。由于XML-RPC先于SOAP而生，其实现也比SOAP简单，因此它沿着自己的路进行发展，并且直到如今仍然广受欢迎。Apache XML-RPC Java库（可以从<http://ws.apache.org/xmlrpc>下载），可以创建使用XML-RPC的Web服务和客户端。

而SOAP是一种更为复杂的Web服务协议，其支持的客户端/服务交互的范围更为广泛。

问：为什么com.java24hours.ws包与Internet主机ws.java24hours.com关联到一起？它们是如何关联的？

答：Java包的名字是由开发该包的程序员创建的。Oracle在给Java类库中的Java包命名时，使用java或javax打头，比如java.util和javax.swing。当其他程序员创建包时，他们遵循了一个惯例，以防止两个实体选择相同的包名称，以及防止两个实体相互混淆。

该惯例是，基于实体所拥有的域名（并将域名前后调换）来选择包的名字。由于我是域名cadenhead.org的拥有者，因此在创建Java类时，其包名可以以org.cadenhead开头，比如org.cadenhead.web。Apache软件基金会（Apache Software Foundation）拥有apache.org域名，因此可以将它的XML-RPC包命名为org.apache.xmlrpc。

22.8 测验

22.8.1 问题

通过回答下述问题来测试你掌握了多少Web服务相关的知识。

1. 服务实现Bean是什么？

- a.** 一个接口，可以识别通过Web服务进行访问的方法。
- b.** 实现了Web服务的类。

- c. 位于Web服务和调用该服务的客户端之间的服务契约。
 - 2. 当像@WebMethod或@Override这样的文本出现在方法声明中时，将调用什么？
 - a. 注解。
 - b. 断言。
 - c. 恼怒。
 - 3. WSDL代表什么？
 - a. Web服务部署语言（Web Services Deployment Language）。
 - b. Web服务描述语言（Web Services Description Language）。
 - c. Lucy in the Sky with Diamonds。

22.8.2 答案

- 1. b. 答案a指的是服务端点接口。
- 2. a. 尽管我觉得答案c也可能是真的，但是这取决于在这一节遇到了多大的麻烦。
- 3. b. WSDL通常被错误地称为Web服务定义语言（Web Services Definition Language）。

22.9 练习

为了进一步巩固本节所学的知识，请做如下练习：

- 在平方根Web服务程序中添加一个方法，该方法将一个数乘以10，然后修改SquareRootClient应用程序来调用该方法。
- 创建一个新的Web服务，它从一个属性文件中读取本地的最高气温、最低气温以及天气情况，然后将这些数据发送到调用Web服务的客户端。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

第23章 创建Java2D图形

本章介绍如下内容：

- 设置文本的字体和颜色；
- 设置容器的背景色；
- 绘制直线、矩形和其他图形；
- 绘制GIF和JPEG图形；
- 绘制填充形状和非填充形状。

在本章，读者将学到如何将Swing容器（存放GUI组件的纯灰色面板和框架）变成一块可以在上面绘制字体、颜色、形状和图形的艺术画布。

23.1 使用Font类

在Java中，颜色和字体使用java.awt包中的Color和Font类来表示。通过使用这些类，你可以用不同的字体和字号来显示文本，也可以改变字体和图形的颜色。字体使用构造函数Font(String, int, int)来创建，该构造函数接收3个参数：

- 字体的名字，它可以是通用名字（比如Dialog、DialogInput、Monospaced、SanSerif或Serif”，也可以是实际的字体名（比如Arial Black、Helvetica或Courier New）；

- `Font.BOLD`、`Font.ITALIC`和`Font.PLAIN`这3个类变量中的其中一个；
- 字体的大小（字号），单位为磅。

下面的语句创建了一个**Font**对象，其字号为12磅，字体为**Serif**，样式为斜体：

```
Font current = new Font("Serif", Font.ITALIC, 12);
```

如果你使用了一个指定的字体，而不是通用的字体，则该字体必须已经安装在运行程序的计算机中。你可以将字体样式合并起来，如下面的例子所示：

```
Font headline = new Font("Courier New", Font.BOLD + Font.ITALIC, 72);
```

当有了字体之后，可以调用**Graphics2D**组件的**setFont(Font)**方法，将其指定为当前字体。在没有指定其他新的字体之前，后续所有的绘制操作都将使用该字体。下面例子中的语句创建了**Comic Sans**字体对象，并在绘制文本之前将其指定为当前地体：

```
public void paintComponent(Graphics comp) {  
    Graphics2D comp2D = (Graphics2D) comp;  
    Font font = new Font("Comic Sans", Font.BOLD, 15);  
    comp2D.setFont(font);  
}
```

```
comp2D.drawString("Potrzebie!", 5, 50);  
}
```

Java支持消除锯齿功能，因此可以更为平滑地绘制字体和图形，而且它们的外观具有较少的锯齿。为了启用这个功能，必须在Swing中设置渲染提示（rendering hint）。Graphics2D对象具有一个setRenderingHint（int, int）方法，它可以接收两个参数：

- 渲染提示键（key）；
- 与该键相关联的值。

这些值是位于java.awt包中的RenderingHints类的类变量。要启用消除锯齿功能，可使用两个参数来调用setRenderingHint()方法：

```
comp2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,  
    RenderingHints.VALUE_ANTIALIAS_ON);
```

在该例子中，comp2D对象是Graphics2D对象，表示一个容器的绘制环境。

23.2 使用Color类

在Java中，颜色可使用Color类表示，其中包括如下常量：black、blue、cyan、darkGray、gray、green、lightGray、magenta、orange、

pink、red、white和yellow。

在容器中，可以调用setBackground(Color)方法来设置组件的背景色，其中上面提到的颜色常量作为该方法的参数，如下例所示：

```
setBackground(Color.orange);
```

与当前的字体相同，在使用setColor(Color)方法执行绘制任务之前，必先设置当前的颜色。在下面的代码中，一条语句将当前颜色设置为orange，并使用该颜色来绘制文本：

```
public void paintComponent(Graphics comp) {  
    Graphics2D comp2D = (Graphics2D) comp;  
    comp2D.setColor(Color.orange);  
    comp2D.drawString("Go, Buccaneers!", 5, 50);  
}
```

在使用setBackground()方法时，我们可以直接在容器上调用，但是setColor()则不行，我们必须在Graphics2D对象上调用该方法。

23.3 创建自定义颜色

在Java中，通过指定通用色彩标准（sRGB）的值，可以创建自定义颜色。sRGB使用颜色中的红、绿、蓝分量来定义颜色。每一种颜色

的取值在0~255之间（0表示没有这种颜色，255表示该颜色分量具有最大值）。

构造函数Color(int, int, int)接收3个参数，这3个参数分别表示红、绿、蓝的值。下面的代码将绘制一个面板，该面板在暗红色（红：235；绿：50；蓝：50）背景中显示亮橙色（红：230；绿：220；蓝：0）文本：

```
import java.awt.*;
import javax.swing.*;

public class GoBucs extends JPanel {
    Color lightOrange = new Color(230, 220, 0);
    Color darkRed = new Color(235, 50, 50);

    public void paintComponent(Graphics comp) {
        Graphics2D comp2D = (Graphics2D) comp;
        comp2D.setColor(darkRed);
        comp2D.fillRect(0, 0, 200, 100);
        comp2D.setColor(lightOrange);
        comp2D.drawString("Go, Buccaneers!", 5, 50);
    }
}
```

该示例调用Graphics2D的fillRect()方法使用当前颜色绘制一个填充的矩形。

***By the
Way***

注意

通过使用sRGB值，可以创建1650万种颜色，不过大多数计算机显示器只能近似地显示其中的大部分颜色。

23.4 绘制直线和形状

在Java程序中，绘制直线和矩形这样的形状就像显示文本那样简单。你只需要一个Graphics2D对象来定义绘制平面和表示要绘制内容的对象即可。

Graphics2D对象有用来绘制文本的方法，如下所示：

```
comp2D.drawString("Draw, pardner!", 15, 40);
```

这将在坐标点(15,40)位置绘制文本“Draw, pardner!”。绘制直线的方法所使用的坐标系与绘制文本的方法使用的相同。(0, 0)坐标位于容器的左上角，当向右移动时，x值增加，当向下移动时，y值增大。通过使用下面的语句，可以确定在框架中或在其他容器中能够使用的最大(x, y)值：

```
int maxXValue = getSize().width;  
int maxYValue = getSize().height;
```

除了可以绘制直线以外，你还可以绘制填充的形状或者是未填充的形状。所谓填充形状就是绘制该形状时，使用当前颜色将该形状空间完全填充起来。而非填充的形状则是只使用当前的颜色绘制的形状的边界。

23.4.1 绘制直线

在创建一个对象的2D图形时，它表示的是正在绘制的形状。

定义形状的对象属于java.awt.geom包的类。

Line2D.Float类能够创建一条连接起点(x, y)和终点(x,y)的直线。下面的语句将创建一条起点为(40,200)，终点为(70,130)的直线：

```
Line2D.Float line = new Line2D.Float(40F, 200F, 70F, 130F);
```

By the Way

注意

Line2D.Float类与之前用到的大多数类不同，它的类名中间有一个句点。这是因为Float是Line2D类中的内部类。有关内部类的内容，请见第20章。

在上面的语句中，参数后面跟有字母F，用来指示参数是浮点型数值。如果省略该字母，则Java会把参数当作整型数值。

除了直线之外，调用Graphics2D类的方法还可以绘制其他形状：`draw()`方法可以绘制轮廓线，而`fill()`方法可以绘制填充形状。

下面的语句可以绘制前面示例中创建的`line`对象：

```
comp2D.draw(line);
```

23.4.2 绘制矩形

矩形可以是填充或非填充的，还可以是圆角的或直角的。矩形可以使用构造函数`Rectangle2D.Float(int, int, int, int)`来创建，创建时指定如下4个参数：

- 矩形左上角的x坐标。
- 左上角的y坐标。
- 矩形的宽度。
- 矩形的高度。

下面的语句可以绘制一个非填充的直角矩形：

```
Rectangle2D.Float box = new Rectangle2D.Float(245F, 65F, 20F,  
10F);
```

这条语句创建的矩形的左上角坐标为(245, 65)，宽度为20，高度为10，宽度和高度都以像素为单位。要绘制该矩形的轮廓，可使用下面的语句：

```
comp2D.draw(box);
```

要填充该矩形，可以使用fill()方法：

```
comp.fill(box);
```

可以使用RoundRectangle2D.Float类来创建圆角矩形。

这个类的构造函数的前4个参数与Rectangle2D.Float类相同，并增加了下面2个参数：

- 水平方向上离矩形角的像素数；
- 垂直方向上离矩形角的像素数。

这些距离用于指定矩形圆角的起始位置。

下面的语句创建一个圆角矩形：

```
RoundRectangle2D.Float ro = new RoundRectangle2D.Float(  
    10F, 10F,  
    100F, 80F,
```

```
15F, 15F);
```

该矩形的左上角坐标为(10, 10)。第3个和第4个参数指定矩形的宽度和高度。在本例中，矩形的宽度为100像素，高度为80像素。

最后两个参数指定在矩形的4个角上，在离角点15像素处开始倒圆角。

23.4.3 绘制椭圆和圆

椭圆和圆是使用同一个类——`Ellipse2D.Float`创建的。这个类的构造函数接收4个参数：

- 椭圆的x坐标；
- 椭圆的y坐标；
- 椭圆的宽度；
- 椭圆的高度。

读者可能已经猜到， (x, y) 坐标不是椭圆或圆的圆心。相反， (x, y) 坐标、宽度、高度描述了椭圆的外接矩形， (x, y) 是该矩形的左上角坐标。如果宽度和高度相同，椭圆将变成圆。

下面的语句创建一个圆，其外接矩形的左上角坐标为 (245, 45)，宽度和高度都是5像素：

```
Ellipse2D.Float cir = new Ellipse2D.Float(  
    245F, 45F, 5F, 5F);
```

23.4.4 绘制弧线

在Java中可绘制的另一个圆形形状是弧线，它是椭圆或圆的一部分。弧线用Arc2D.Float类创建，这个类的构造函数使用几个与Ellipse2D.Float相同的参数。要创建弧线，需要指定一个椭圆、该椭圆的可见部分（单位为度）以及弧线的起点。

要创建弧线，向构造函数传递下述整型参数：

- 椭圆外接矩形左上角的x坐标；
- 该矩形左上角的y坐标；
- 该矩形的宽度；
- 该矩形的高度；
- 弧线的起点（0°~359°）；
- 弧线长度，单位为度；
- 弧线类型。

弧线的起点为0°~359°，方向为逆时针方向，其中0°对应于3点钟的位置，如图23.1所示。

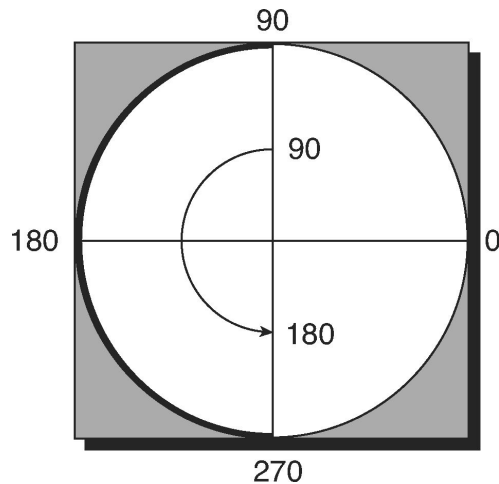


图23.1 如何使用度数来定义弧线

弧线的类型用3个类变量之中的一个指定：**PIE**将弧线绘制为饼图的一部分；**CLOSED**将弧线的起点和终点用直线相连；**OPEN**不将终点和起点相连。

下面的语句创建一个非闭合弧线，其外接矩形的左上角坐标为（100, 50），宽度为65，高度为75，弧线从30°处开始，长120°：

```
Arc2D.Float smile = new Arc2D.Float(100F, 50F, 65F, 75F,  
    30F, 120F, Arc2D.Float.OPEN);
```

23.5 绘制饼图

在结束本章前，将创建图形用户界面组件**PiePanel**，它显示一个饼图。该组件是**JPanel**的子类，**JPanel**是一个简单的**Swing**容器，非常适合用于在其中绘制内容。

开始创建类之前，应定义创建其对象的方式。使用**PiePanel**类的应用程序必须执行下面的步骤：

- 使用构造函数**PiePanel**（**int**）创建一个**PiePanel**对象，其中的整型参数指定饼图包含的切片数；
- 调用对象的**addslice**（**color, float**）方法给切片指定颜色和值。

PiePanel中的每个切片的值为其表示的数量。

例如，表23.1列出了在头38年中，美国学生还贷数据，这些数据是高等教育办公室提供的。

表23.1 美国学生还贷统计表

已归还的贷款总额	1010亿美元
在校生贷款总额	680亿美元
正在归还的贷款总额	910亿美元
拖欠贷款总额	250亿美元

可以使用**PiePanel**在饼图中表示这些数据，为此可使用下述代码：

```
PiePanel loans = new PiePanel(4);
loans.addSlice(Color.green, 101F);
loans.addSlice(Color.yellow, 68F);
loans.addSlice(Color.blue, 91F);
loans.addSlice(Color.red, 25F);
```


图23.2所示为一个应用程序运行时显示出的结果，该应用程序包含了一个使用学生贷款数据创建的PiePanel组件。

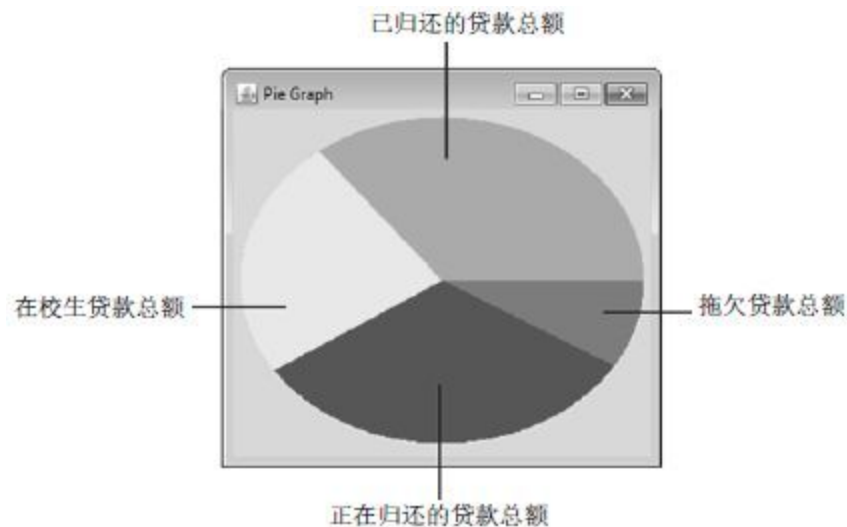


图23.2 在饼图中显示学生贷款数据

在创建PiePanel对象时，构造函数中指定了切片数。要绘制每个切片，还需要知道其他3项内容：

- 切片颜色，用Color对象表示；
- 每个切片代表的值；
- 所有切片代表的总值。

使用辅助类PieSlice来表示饼图中的每个切片：

```
import java.awt.*;  
  
class PieSlice {  
    Color color = Color.lightGray;
```

```
float size = 0;

PieSlice(Color pColor, float pSize) {
    color = pColor;
    size = pSize;
}
}
```

每个切片都是通过调用**PieSlice(Color, float)**创建的。所有切片的总值将存储在**PiePanel**类的私有实例变量**totalSize**中，这个类中还有实例变量**background**和**current**，它们分别用于存储面板背景色和对切片进行计数：

```
private int current = 0;
private float totalSize = 0;
private Color background;
```

有了**PieSlice**类后，就可以使用另一个实例变量创建一个**PieSlice**对象数组：

```
private PieSlice[] slice;
```

创建**PiePanel**对象时，没有指定切片的颜色和大小。在构造函数中只需指定**slice**数组的长度并保存面板的背景色：

```
public PiePanel(int sliceCount) {
    slice = new PieSlice[sliceCount];
    background = getBackground();
}
```

`addSlice(Color, float)`方法用于在面板中添加饼图切片:

```
public void addSlice(Color sColor, float sSize) {
    if (current <= slice.length) {
        slice[current] = new PieSlice(sColor, sSize);
        totalSize += sSize;
        current++;
    }
}
```

`current`实例变量用于将每个切片存储到`slice`数组的相应元素中。数组的`length`变量指出了数组被定义为存储多少个元素，因此只要`current`不大于`slice.length`，就可以继续往面板中添加切片。

正如读者预期的，`PiePanel`类在其`paintComponent()`方法中处理所有的图形操作。该任务中最棘手的是绘制代表每个饼图切片的弧线。

这是由下面的语句处理的:

```
float start = 0;
for (int i = 0; i < slice.length; i++) {
    float extent = slice[i].size * 360F / totalSize;
    comp2D.setColor(slice[i].color);
    Arc2D.Float drawSlice = new Arc2D.Float(
        xInset, yInset, width, height, start, extent,
```

```
        Arc2D.Float.PIE);
    start += extent;
    comp2D.fill(drawSlice);
}
```

变量`start`用于记录弧线的起点，变量`extent`用于记录弧线的长度。如果知道所有饼图切片的总值和每个切片的值，便可以将切片的值乘以360再除以所有切片的总值，从而计算得到`extent`的值。

所有弧线都是在一个`for`循环中绘制的：计算出每个弧线的`extent`，创建该弧线并将`start`增加`extent`。这确保下一个切片紧邻最后一个切片。最后，调用`Graphics2D`的`fill()`方法绘制弧线。

要将上述内容组合在一起，创建一个名为`PiePanel`的Java空文件，然后输入程序清单23.1所示的内容。

程序清单23.1 PiePanel.java的完整源代码

```
1: package com.java24hours;
2:
3: import java.awt.*;
4: import javax.swing.*;
5: import java.awt.geom.*;
6:
7: public class PiePanel extends JPanel {
8:     private PieSlice[] slice;
9:     private int current = 0;
10:    private float totalSize = 0;
11:    private Color background;
12:
13:    public PiePanel(int sliceCount) {
14:        slice = new PieSlice[sliceCount];
15:        background = getBackground();
16:    }
17:
```

```

18:     public void addSlice(Color sColor, float sSize) {
19:         if (current <= slice.length) {
20:             slice[current] = new PieSlice(sColor, sSize);
21:             totalSize += sSize;
22:             current++;
23:         }
24:     }
25:
26:     public void paintComponent(Graphics comp) {
27:         super.paintComponent(comp);
28:         Graphics2D comp2D = (Graphics2D) comp;
29:         int width = getSize().width - 10;
30:         int height = getSize().height - 15;
31:         int xInset = 5;
32:         int yInset = 5;
33:         if (width < 5) {
34:             xInset = width;
35:         }
36:         if (height < 5) {
37:             yInset = height;
38:         }
39:         comp2D.setColor(background);
40:         comp2D.fillRect(0, 0, getSize().width,
41: getSize().height);
42:         comp2D.setColor(Color.lightGray);
43:         Ellipse2D.Float pie = new Ellipse2D.Float(
44:             xInset, yInset, width, height);
45:         comp2D.fill(pie);
46:         float start = 0;
47:         for (int i = 0; i < slice.length; i++) {
48:             float extent = slice[i].size * 360F / totalSize;
49:             comp2D.setColor(slice[i].color);
50:             Arc2D.Float drawSlice = new Arc2D.Float(
51:                 xInset, yInset, width, height, start,
52: extent,
53:                 Arc2D.Float.PIE);
54:             start += extent;
55:             comp2D.fill(drawSlice);
56:         }
57:     }
58:
59:     class PieSlice {
60:         Color color = Color.lightGray;
61:         float size = 0;
62:
63:         PieSlice(Color pColor, float pSize) {
64:             color = pColor;
65:             size = pSize;
66:         }

```

```
66: }
```

在程序清单23.1中，第1~56行定义了PiePanel类，在第58~66行定义了PieSlice辅助类。PiePanel类可以在任何Java程序的GUI中用作组件。要测试PiePanel，需要创建一个类，并在其中使用该PiePanel。

程序清单23.2是一个使用PiePanel类的应用程序—PieFrame。创建一个Java空文件，然后输入程序清单23.2中的所有文本。

程序清单23.2 PieFrame.java的完整源代码

```
1: package com.java24hours;
2:
3: import javax.swing.*;
4: import java.awt.*;
5:
6: public class PieFrame extends JFrame {
7:     Color uneasyBeingGreen = new Color(0xCC, 0xCC, 0x99);
8:     Color zuzusPetals = new Color(0xCC, 0x66, 0xFF);
9:     Color zootSuit = new Color(0x66, 0x66, 0x99);
10:    Color sweetHomeAvocado = new Color(0x66, 0x99, 0x66);
11:    Color shrinkingViolet = new Color(0x66, 0x66, 0x99);
12:    Color miamiNice = new Color(0x33, 0xFF, 0xFF);
13:    Color inBetweenGreen = new Color(0x00, 0x99, 0x66);
14:    Color norwegianBlue = new Color(0x33, 0xCC, 0xCC);
15:    Color purpleRain = new Color(0x66, 0x33, 0x99);
16:    Color freckle = new Color(0x99, 0x66, 0x33);
17:
18:    public PieFrame() {
19:        super("Pie Graph");
20:        setLookAndFeel();
21:        setSize(320, 290);
22:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23:        setVisible(true);
24:
25:        PiePanel pie = new PiePanel(10);
26:        pie.addSlice(uneasyBeingGreen, 1350);
27:        pie.addSlice(zuzusPetals, 1221);
```

```

28:         pie.addSlice(zootSuit, 316);
29:         pie.addSlice(sweetHomeAvocado, 251);
30:         pie.addSlice(shrinkingViolet, 201);
31:         pie.addSlice(miamiNice, 193);
32:         pie.addSlice(inBetweenGreen, 173);
33:         pie.addSlice(norwegianBlue, 164);
34:         pie.addSlice(purpleRain, 143);
35:         pie.addSlice(freckle, 127);
36:         add(pie);
37:     }
38:
39:     private void setLookAndFeel() {
40:         try {
41:             UIManager.setLookAndFeel(
42: "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
43:         );
44:         } catch (Exception exc) {
45:             // ignore error
46:         }
47:     }
48:
49:     public static void main(String[] arguments) {
50:         PieFrame pf = new PieFrame();
51:     }
52: }

```

PieFrame类是一个简单的图形用户界面，包含一个组件，该组件是在第25行创建的**PiePanel**对象。在第26~35行，该对象的**addSlice()**方法被调用了10次，以将切片添加到饼图中。

运行该应用程序时，**PieFrame**将显示一个饼图，其中包含10个人口最多的国家的人口数量（单位为百万），这些数据来自美国人口普查局于2013年12月发表的国际数据报告。按从多到少的顺序依次为：中国（13.50亿）、印度（12.21亿）、美国（3.16亿）、印尼（2.51亿）、巴西（2.01亿）、巴基斯坦（1.93亿）、尼日利亚（1.73亿）、孟加拉国（1.64亿）、俄罗斯（1.43亿）和日本（1.27亿）。

由于Java在Color类中只定义了几种颜色，因此这里创建了10种新颜色，并赋予它们描述性名称。这些颜色是用十六进制值表示的（在Java中，十六进制值以0x打头），但也可以在Color()构造函数使用十进制值来指定它们。

图23.3所示为运行该应用程序时的结果。

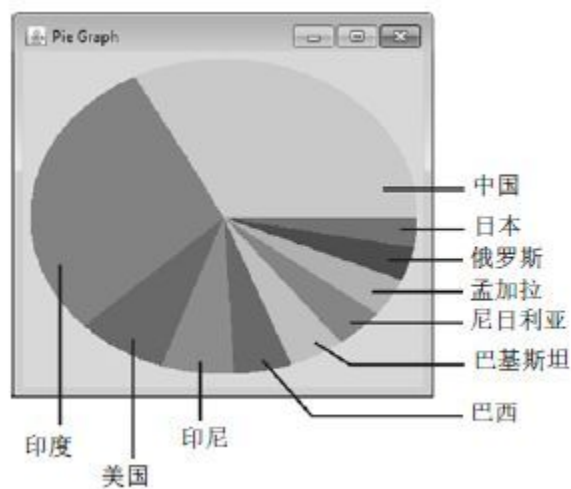


图23.3 在饼图中显示人口数据

By the Way

注意

通过访问www.cadenhead.org/census，可以找到美国人口普查局发布的当前国际人口数据。

23.6 总结

通过使用字体、颜色和图形，可以让程序中的元素更具备吸引力，从而引起用户的注意。

尽管使用Java中的形状在进行绘制时，看起来相当麻烦，而且不是那么值得，但是，与从图像文件中载入的图形相比，使用多边形描绘的图形具有两个优势。

- 速度：即使是加载和显示小图形（比如图标），所需的时间也比加载和显示一系列多边形长。
- 缩放：对于由多边形组成的图像，只需修改用于创建多边形的值就能改变整个图像的大小。例如，可在Sign类中添加一个函数，将每个形状的(x, y)点乘以2再创建它们，这样图像将比原来大一倍。缩放多边形图像的速度比图像文件快，且得到的结果更好。

23.7 问与答

问：如何沿顺时针方向（而不是逆时针方向）绘制弧线？

答：可以将弧线的长度设置为负数。弧线起点不变，但方向相反。例如，下面的语句绘制一条这样的非闭合弧线，其外接矩形的左上角坐标为（35, 20），高度为20，宽度为15，弧线的长度为90°，起点为0°，方向顺时针方向：

```
Arc2D.Float smile = new Arc2D.Float(35F, 20F, 15F, 20F,  
0F, -90F, Arc2D.Float.OPEN);
```

问：椭圆和圆没有角。在构造函数`Ellipses.Float`中指定的 (x, y) 坐标是什么？

答： (x, y) 坐标是椭圆和圆中最小 x 坐标和最小值 y 坐标，如果绘制一个外接椭圆或圆的矩形，该矩形左上角的 (x, y) 坐标就是该方法中指定的参数。

问：如何在Java中使用XRender？

答：在基于X11的环境中（通常是Linux），Java 支持使用XRender渲染引擎来绘制Java2D图形。该功能在默认情况下是关闭的，必须使用命令行选项`-Dsun.java2d.xrender=true`来启用该功能。XRender允许Java程序使用现代图形处理单元（Graphics Processing Unit, GPU）的性能来绘制图形。

在NetBeans中，通过选择Run->Set Project Configuration->Customize可以设置该选项。使用VM Options字段来设置该选项，然后单击OK即可。

23.8 测验

回答下面的问题，看是否掌握了字体和颜色的使用技巧。

23.8.1 问题

1. 下面哪一个不是用来选择颜色的常量？

a. `Color.cyan`。

b. `Color.teal`。

c. `Color.magenta`。

2. 当对颜色进行更改并在容器中进行重绘时，必须怎么做才能使其可见？

a. 使用`drawColor()`方法。

b. 使用`repaint()`语句。

c. 什么都不做。

3. RGB表示什么意思？

a. Roy G. Biv。

b. Red Green Blue。

c. Lucy in the Sky with Diamonds。

23.8.2 答案

1. **b.** Jacksonville Jaguars的原色teal已经从`Color`类中移除。

2. **b.** 调用`repaint()`方法时，必须手动调用`paintComponent()`方法。

3. **b.** 如果c选项是正确答案，则只有在若干年后回忆时，才会“看到”（钻石）的颜色。

23.9 练习

为了进一步巩固你在编程中使用字体和颜色的技巧，请完成下面的练习。

- 创建**PieFrame**类的一个新版本，其中，颜色值和饼图切片的价值不再出现在应用程序的源代码中，而是作为命令行参数传入。
- 创建一个应用程序，它使用颜色、形状和字体在面板中绘制一个“停止”标记。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站//www.java24hours.com。

第24章 编写Android app

本章介绍如下内容：

- 为什么要创建Android；
- 创建Android app；
- Android app是如何架构的；
- 在模拟器上运行app；
- 在Android手机上运行app。

Java是一款通用的编程语言，可以在多种平台上运行。其中有一个平台在最近几年发展势头迅猛，成为使用Java语言进行开发的一个全新领域。

Android最初作为在手机上使用的操作系统，如今已经应用到其他设备中，并专门运行使用Java编写的程序。

这些称为app的程序是在开源的移动平台上开发的，该平台对开发者来说完全免费。任何人都可以编写、部署和销售Android app。

在本章，你将学习到Android是如何出现的，它为什么这么特殊，以及为什么数以万计的程序员在该平台上进行开发。通过本章的学习，你还可以创建一个app，并将其运行在Android手机上。如果没有Android手机，则可以将其运行在模拟器上。

24.1 Android简介

Google在2005年收购了Android技术，并于两年后将其推出，旨在通过业界的努力来建立一种全新而且开放的移动手机平台，该平台没有专利保护，因此与RIM黑莓和苹果iPhone所使用的技术不同。一些著名的移动手机生产厂商和技术厂商，包括Google、Intel、Motorola、Nvidia、Samsung以及其他公司在内，组成了开放手机联盟（Open Handset Alliance），旨在本着互惠互利的目的，来推动这个新平台的发展。

Google发布了Android软件开发工具包（SDK），这是一个免费的工具集，用以开发Android app。运行Android的第一款手机是T-Mobile G1，它于2008年6月上市。

Android技术在发展初期进展缓慢，但是从2010年初期开始呈爆发态势增长，并成为iPhone和其他移动平台的强劲对手。所有的主流手机运营商现在都提供Android手机，而且Android在平板电脑和电子书阅读器市场也得到了显著增长。

在Android出现之前，如果要进行移动应用程序的开发，需要昂贵的编程工具和开发计划。而且手机生产厂商可以决定谁能够为其生产的手机开发app，还可以决定app是否可以卖给其他用户。

Android的出现打破了这一壁垒。

Android的开源和非专利属性意味着任何人都可以开发、发布和销售app。而且其中涉及的唯一成本是将app提交给Google app市场时收取的象征性费用，其他的都是免费的。

在Android开发者站点 (<http://developer.android.com>) 上可以下载Android SDK, 以及其他编程工具。当你开发app时, 将会经常求助于该站点, 因为它囊括了Android Java类库中的所有类, 因此可以作为详尽的在线资源进行参考。

如果你使用的IDE支持Android SDK, 则可以很容易地开发Android app。最流行的Android编程IDE是Eclipse, 它也是免费而且开源的。Eclipse的一个Android插件可以使得Android SDK功能无缝地嵌入到Eclipse中。

你可以使用Eclipse来编写Android app, 并在模拟器 (其运行行为很像Android手机的软件) 上进行测试, 甚至也可以将app部署在真实的Android设备上。

在过去, Java语言主要用来编写运行在3个场合的程序, 这3个场合是台式计算机、Web服务器和Web浏览器。

Android可以让Java用在各处。你用Java编写的程序可以部署在上百万台手机和其他移动设备上。

在20世纪90年代中期, James Gosling发明Java语言的初衷就是希望该语言能够运行在各种设备上, 比如手机、智能卡和家电中。如今, Android的出现实现了Java语言的设计初衷。

当Java语言先是作为运行交互式Web程序的一种方式, 继而成为通用编程语言, 而逐渐流行起来时, Java开发人员已经将Java的设计初衷抛到一边。

20年过后，据业内人士估计，Android平台承载了全世界十几亿的Java程序。

随着时间的发展，Android将成为最普遍、最具潜力的Java编程领域，而且一定会是硕果累累。

Watch Out!

警告：

本章内容在该书中占据的篇幅最长，因为当你作为Android app开发人员起步，朝着将来的百万富翁努力时，需要知晓很多内容。

24.2 创建Android app

Android app是最常见的Java程序，它使用了应用程序的框架，该框架是所有app共有的一组核心类和文件。为了让app在Android设备上正确运行，该框架包含了一组app架构规则。

要开始编写app，你必须安装并配置Android SDK、Eclipse IDE，以及用于Eclipse的Android插件。

如果你是头一次进行Android编程，你可以在附录D找到获取并安装这些工具的方法。

将要创建的第一个项目是编写一个SalutonMondo app，该程序可以在Android设备的屏幕上显示一行文本。

1. 运行Eclipse IDE，它的外观和行为与NetBeans很像。
2. 选择File->New->Android Project，打开New Android Application向导，如图24.1所示。



图24.1 在Eclipse中创建一个新的Android项目

3. 在Application Name文本框中输入SalutonMondo。该名字将自动输入到Project Name文本框中。

4. Package Name文本框应该包含这个app所属的Java包的名字。在该文本框中输入org.cadenhead.android。

5. 每一个Android项目需要一个构建目标（build target）。该目标表示可以运行app的Android最老版本。由于每一个新发布的Android版本都有增强的特性，你选择的目标决定了可以使用的特性。

在Minimum Required SDK和 Target SDK下拉列表中选择API 19。

6. 单击Next按钮。该向导会咨询与项目相关的更多问题。

7. 此时在工作区中应该有3个复选框按钮被选中：Create Custom Launcher Icon、Create Activity和Create New Project（如果没有选中，则将其选中）。

8. 单击Next。该向导会问应用程序图标相关的问题。

9. 可以接受图标的默认设置，因此单击Next。接下来将询问你要创建的行为。

10. 选择Fullscreen Activity然后单击Next。

11. 行为（activity）是app可以完成的任务。在Activity Name文本框中输入Saluton Activity。

12. 单击Finish。该app创建完毕，SalutonMondo条目出现在Package Explorer面板中。

24.2.1 剖析一个Android新项目

在一个Android app项目中，大约包含20个文件和文件夹，而且它们的组织方式相同。取决于app的功能，它可以包含更多的文件，但是这些开始文件和文件夹必须存在。

图24.2所示为在创建一个新的Android项目后，Eclipse Package Explorer的界面。

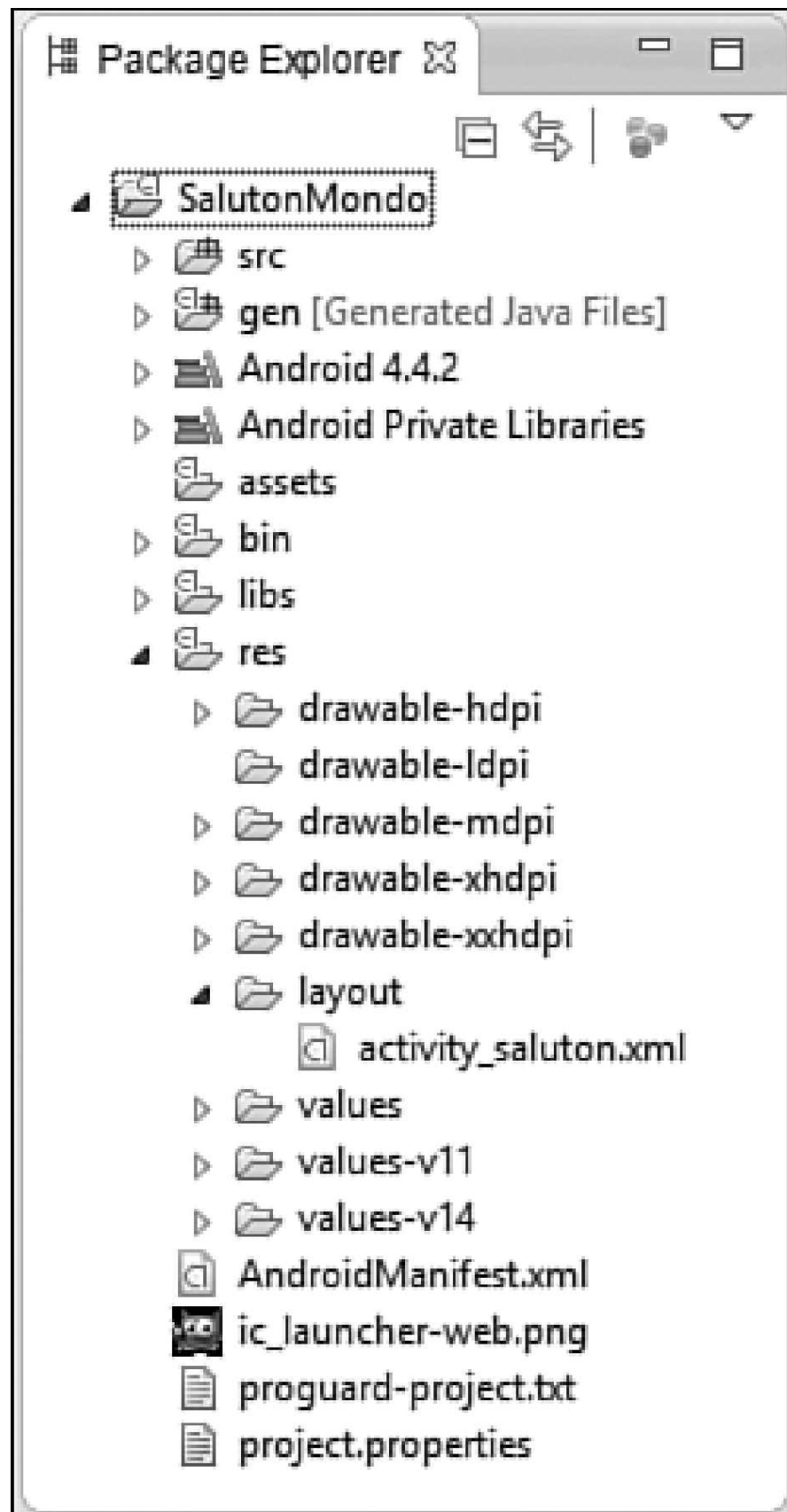


图24.2 查看Android项目的组成部分

你可以使用文件夹来探索该项目的文件和文件夹结构。这个新的 SalutonMondo app在开始时具有如下组件。

- /src文件夹：该app的Java源代码的根文件夹。
- /src/org.cadenhead.android/SalutonActivity.java：在该app运行时，将会默认执行的行为类。
- /gen文件夹：用于存放生成的Java源代码，你不需要手动编辑这些代码。
- /gen/org.cadenhead.android/R.java：为该app自动生成的资源管理源代码（不要手动编辑！）。
- /assets：不会被编译到app中的文件夹或文件资源。
- /res：存放应用程序资源（如字符串、数字、布局文件、图形和动画）的文件夹。还有用于特殊资源类型的子文件夹：layout、3个values文件夹和5个drawable文件夹。
- AndroidManifest.xml：app的主配置文件。
- default.properties：由Android插件生成的构建（build）文件，不要对其进行编辑。
- project.properties：一个配置文件，用于自动生成的项目。不要对其进行手动编辑。

这些文件形成了应用程序的框架。作为Android程序员，你要做的第一件事就是学习如何修改该框架，以便发现每个组件要完成的功能。

该框架中还包含实现特定目的的其他文件。

24.2.2 创建app

尽管到目前为止什么也没有做，但是你可以成功地运行这个Android项目。该项目的框架可以作为一个app运行。

但是这一点也不好玩，因此可以自定义SalutonMondo app，使其显示传统的计算机编程问候语“Saluton Mondo!”。

在第2章，是通过调用System.out.println()方法，并将文本“Saluton Mondo!”作为字符串显示出来的。

Android app显示已经存储在strings.xml资源文件中的字符串。可以在/res/values文件夹中找到该文件。

在Package Explorer中导航到该文件夹。双击strings.xml，打开资源编辑器，如图24.3所示。

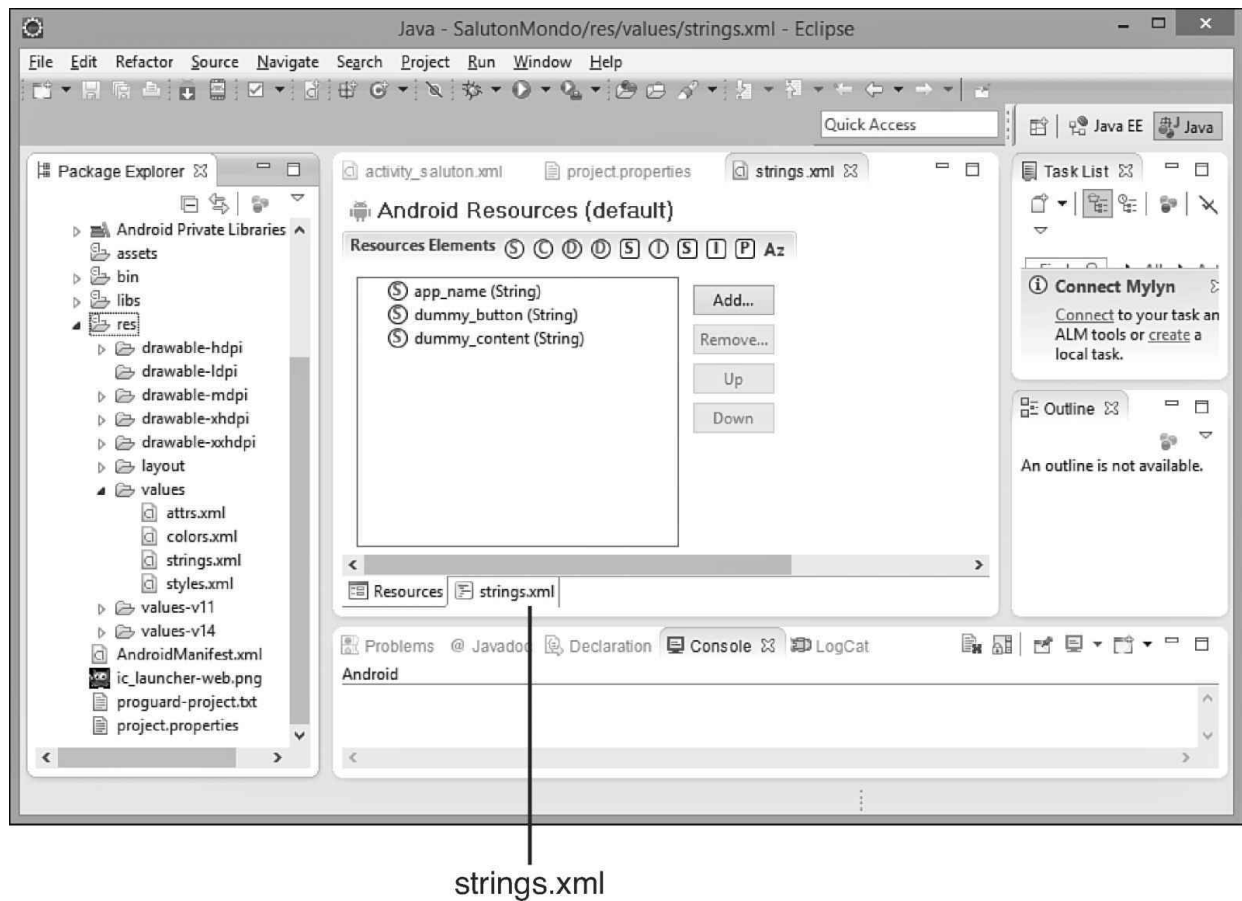


图24.3 编辑Android app的字符串资源

和Java中的变量一样，字符串和其他资源都有名字和值。在资源元素面板中列出了3个字符串资源： `app_name`、`dummy_button`和 `dummy_content`。

在给资源命名时，需要遵循3个规则：

- 必须小写；
- 没有空格；
- 只能使用下划线字符作为标点。

在资源元素面板中单击一个字符串，将会出现出现Name文本框和Value文本框，而且还会带有如何编辑字符串的指导。

在运行New Android Project向导时，需要选择app_name字符串资源。该名字应该与之前命名的名字匹配，但是可以随时修改该字符串。

字符串dummy_content包含app运行时，显示在app主界面（也是唯一的界面）的文本。单击该字符串的名字可以对其修改。

在Value文本框，输入“Saluton Mondo！”。

资源存储在XML文件中。Resources编辑器是一个简单的XML编辑器。你也可以直接编辑XML文件自身。单击编辑器底部的strings.xml选项卡，载入该文件，即可直接进行编辑（该选项卡在图24.3中做出了标识）。

此时，strings.xml看起来如下所示：

```
Output ▼  

```

在该编辑器中可以修改XML文件中的所有内容，甚至是标记。string元素包含一个name属性，表示该资源的名字。该元素的值作为字符数据封装在标记内部。

返回Resources编辑器，单击Resources选项卡。然后在Eclipse工具栏中单击Save按钮来保存对strings.xml文件做出的修改。

修改之后，即可准备运行该app。

Watch Out!

警告：

尽管你可以直接修改XML文件，但最好别修改。在创建Android app的资源时，通常没有必要修改XML文件。但是，如果在定义资源时，Eclipse编辑器对其中的某些内容不支持，则需要修改XML文件。在字符串中不存在这样的问题，所以最好别在Resources编辑器中修改，否则将会带来错误。

24.2.3 安装Android模拟器

在生成（build）Android app之前，必须设置其调试环境，这可以在Eclipse内处理。你必须在台式计算机上安装可以运行app的Android虚拟设备（Android Virtual Device，AVD），将其作为模拟器。你还必须创建项目的调试配置。这一切完成之后，你可以生成该app，并在模拟器中运行它。

为了配置Android虚拟设备，首先在Eclipse工具栏中单击一个绿色的Android电话图标，如图24.4所示。

虚拟设备管理器

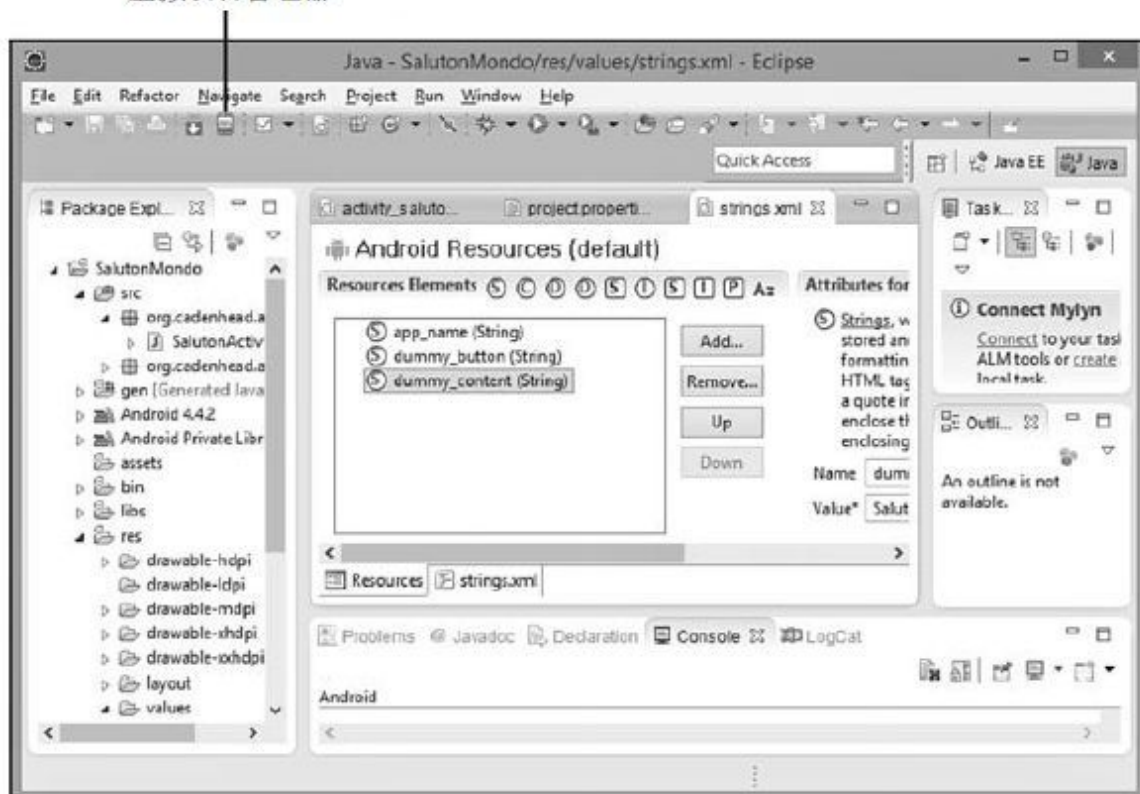


图24.4 配置Android虚拟设备

这将启用Android虚拟设备管理器，这是Android SDK中的一个工具。你创建的模拟器将列在左侧面板中。管理器如图24.5所示。

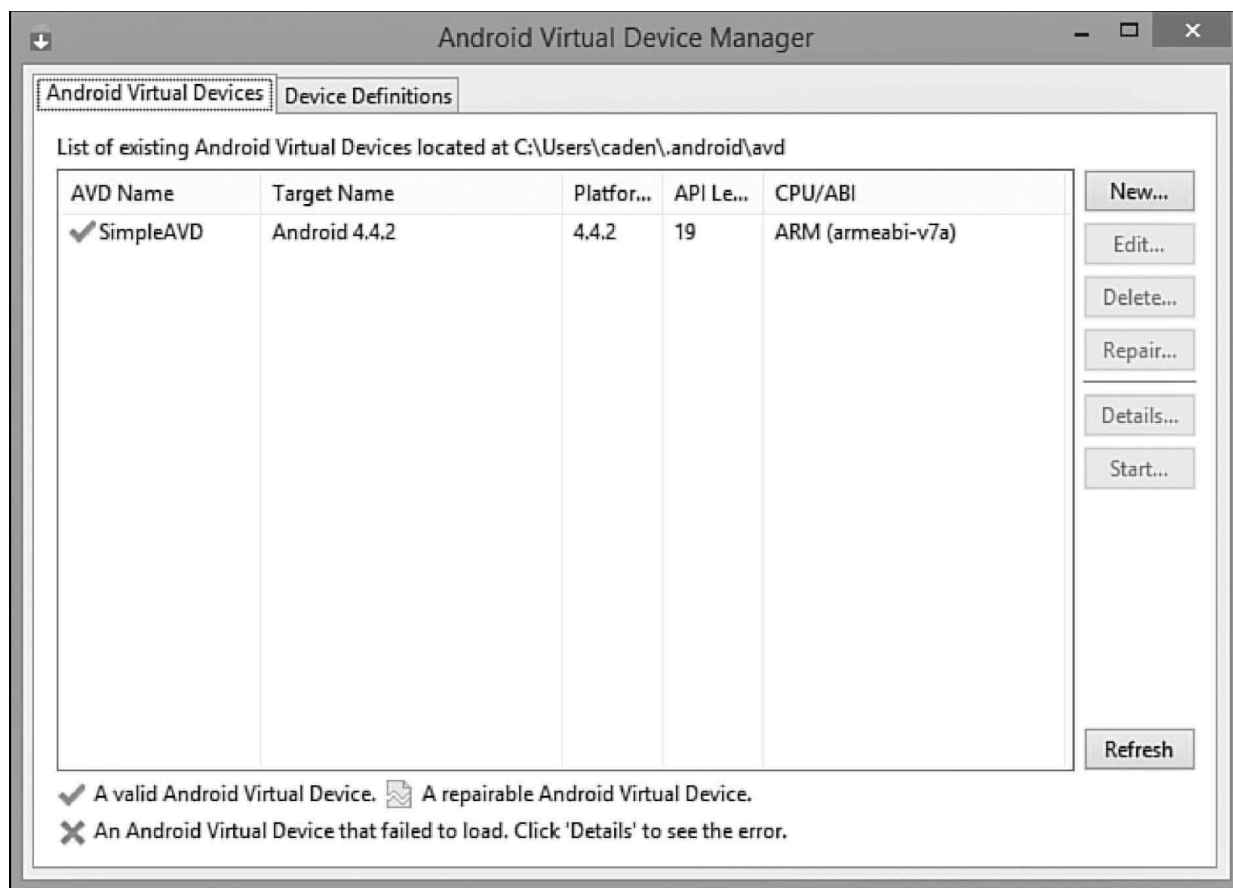


图24.5 创建一个新的Android模拟器

为了增加一个新的模拟器，单击**New**按钮，然后执行如下步骤。

1. 在AVD Name文本框中输入“SimpleAVD”。
2. 在Device文本框中选择一个电话，比如Nexus S。
3. 在Target文本框中，必须选择Android的目标版本。这里选择的是Android 4.4.2–API Level 19。
4. 在Skin字段选择No Skin。

5. 在Memory Options区域，在RAM下输入768（当该数值较大时，在某些Windows计算机上载入模拟器时会引发失败）。

6. 在SD Card区域，在Size文本框中选择模拟SD卡的大小。输入1024然后从后面的下拉列表中选择MiB，这表示SD卡的大小为1024MB。你的计算机上必须有足够可用的空间，如果你不希望占据太大空间，则可以调小该值。最小值为9MB。

7. 单击OK按钮，很快就可以创建一个新的模拟器（通常不会长于1分钟）。

可以根据需要创建多个模拟器。可以对它们进行自定义，使其用于不同的Android版本的显示类型。

关闭Android虚拟设备管理器，返回Eclipse主界面。

24.2.4 创建调试配置

在启用SalutonMondo app之前，需要做的最后一件事是在Eclipse中创建调试配置，其步骤如下。

1. 选择Run->Debug Configuratons，打开Debug Configurations窗口。

2. 在左侧面板中，双击Android Application条目（见图24.6），可以看到新创建了一个New_Configuration条目，该条目是Android Application的子项。右侧面板将显示这个新条目的一些配置选项。

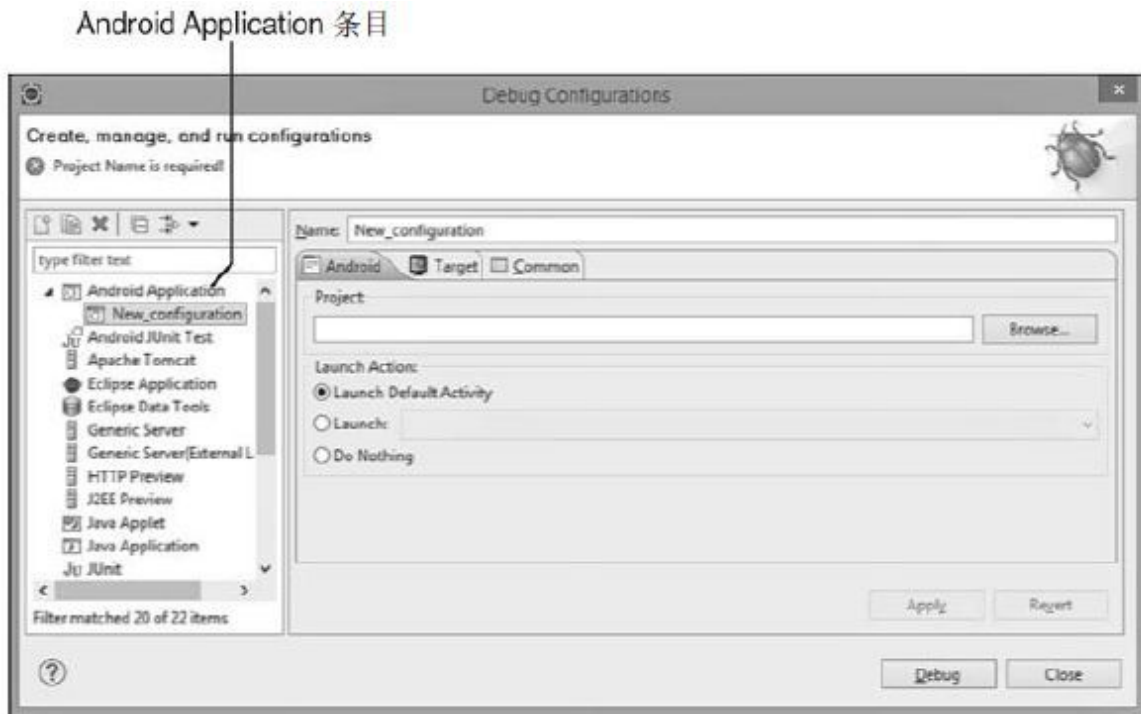


图24.6 创建Android调试配置

3. 在右侧面板中的Name文本框中，将其值修改为SalutonDebug。
4. 单击Browse按钮，打开Project Selection对话框。
5. 选择SalutonMondo项目，然后单击OK。
6. 单击Target选项卡。
7. 在Deployment Target Selection Mode下，选择Automatically Pick Compatible Device（如果之前没有被选择）。然后在表格中选择目标AVD。
8. 在表格中，选择SimpleAVD模拟器复选框。
9. 单击Apply按钮保存所有变更，然后单击Close。

24.3 运行app

现在你已经有了Android模拟器，并创建了调试配置，接下来可以运行这个app了。单击位于Package Explorer顶部的SalutonMondo，然后单击Eclipse工具栏中的调试图标。

Android模拟器将该app载入到它自己的窗口中，这可能需要一分钟甚至更长的时间，所以在它启动的过程中，请耐心等待。

该app显示时，将带有一个Android图标以及SalutonMondo标签。单击Android图标运行它。

该模拟器将“Saluton Mondo！”作为文本和app的标题显示出来，如图24.7所示。模拟器可以像手机那样工作，但是此时需要用鼠标单击按钮（而不是用手）来辅助它的运行。单击“Back”按钮，关闭app，来看一下它是如何模拟Android设备的。

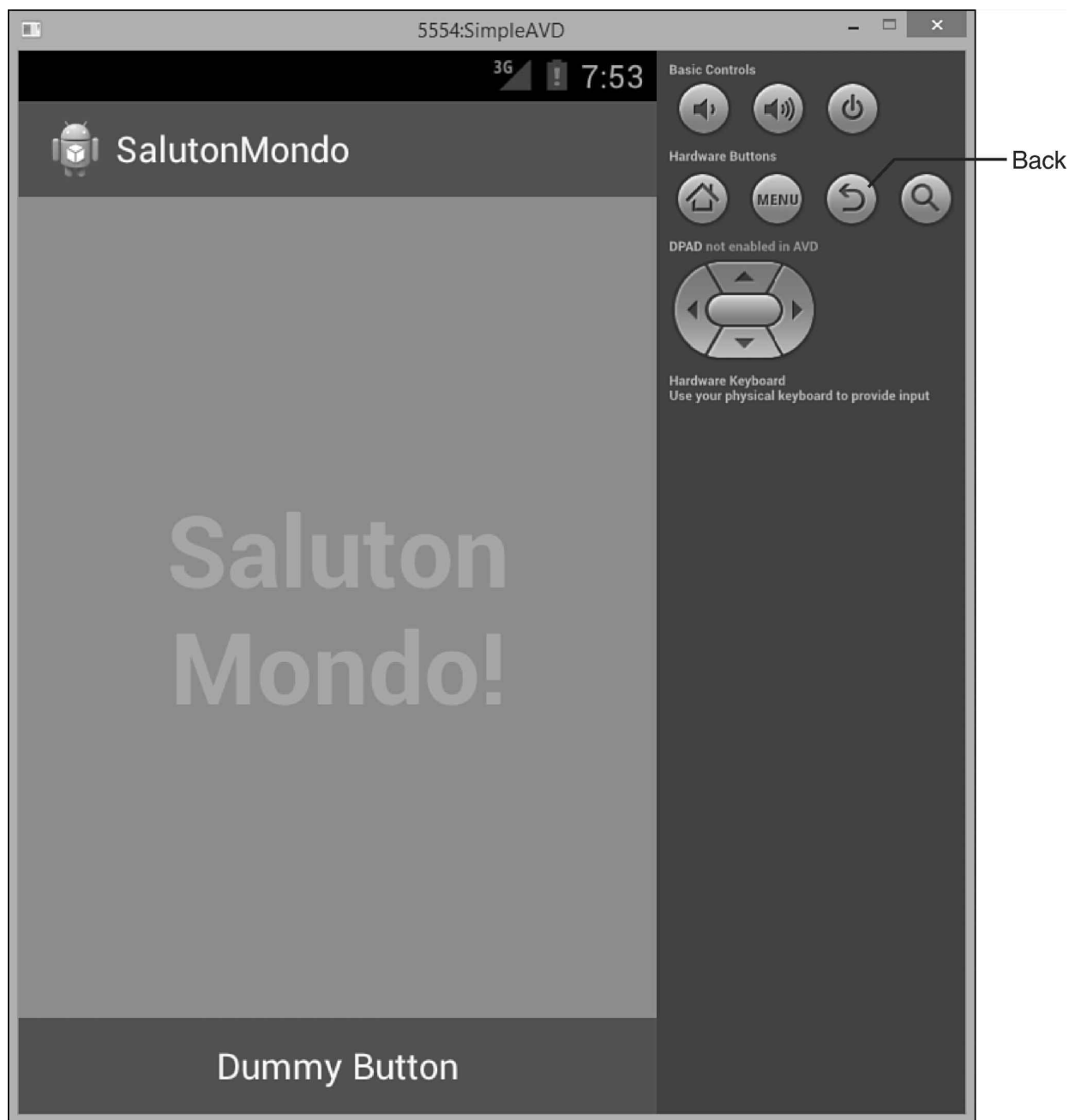


图24.7 在Android模拟器中运行app

和真实设备一样，模拟器可以做很多事情，比如在计算机已经连接到Internet的前提下，模拟器也可以连接到Internet，而且它还可以接收虚假的电话呼叫和短消息。

模拟器毕竟不是功能齐备的设备，因此你开发的app必须要在真实的Android手机和平板电脑上进行测试。

如果可以使用USB线缆将Android手机（或其他设备）连接到计算机，在该手机被设置为调试模式的前提下，你可以在上面运行app。使用Android SDK开发的app只能部署在处于调试模式的手机中。

在手机中，通过选择Home->Settings->Applications->Development（或Home->Settings-> Developer Options），进入调试模式。此时将显示Development设置，从中选择USB debugging选项。

接下来，在Eclipse中执行如下步骤。

1. 选择Run->Debug Configurations，打开Debug Configuration窗口。
2. 在右侧面板中单击Target选项卡。
3. 将Deployment Target Selection Mode改为Always Prompt to Pick Device。
4. 单击Apply和Close。

使用USB线缆将Android手机连接到计算机上。此时将会在屏幕顶部的工具栏中出现一个Android bug图标。如果向下拖动该工具栏，将会看到消息“USB Debugging Connected”。

返回Eclipse，单击工具栏中的bug图标，打开Android Device Chooser对话框（见图24.8）。

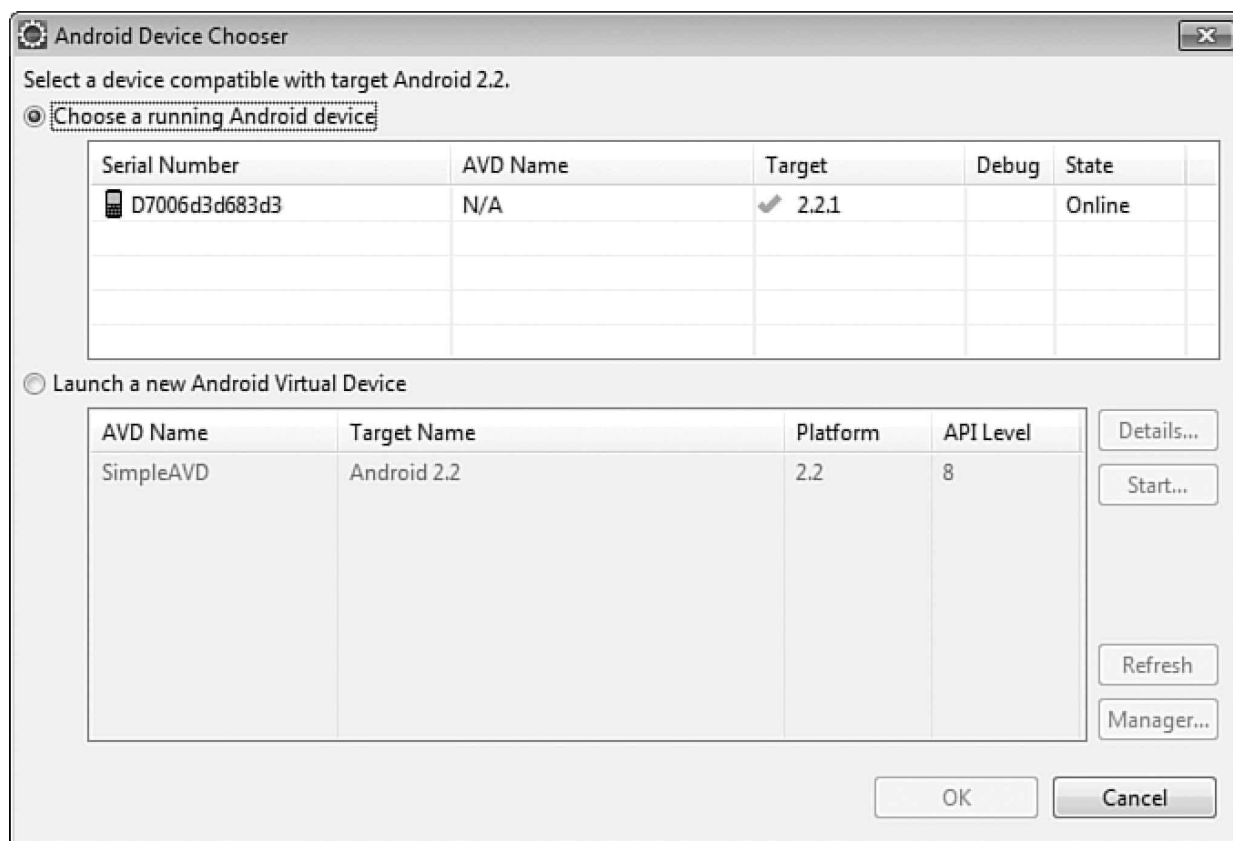


图24.8 在Android手机上部署app

如果Android手机被检测到，它将显示在图24.8中Choose a Running Device选项下面的表格中。

选中Choose a Running Device选项，单击手机设备的名字，单击OK。该app将运行在手机中。

如同在第2章创建的第一个Java程序那样，你创建的第一个Android app非常稀松平常。你接下来将要创建的这个项目将不会如此。

24.4 设计真实的app

Android app可以使用设备的所有功能，比如短消息服务、基于位置的服务、以触摸进行的输入。在本章最后第一个编程项目中，你将创建一个真实的app，名字为Take Me To Your Leader。

该app使用了Android手机的功能来进行电话呼叫、浏览网站，以及使用Google地图进行定位。该app可以让你通过手机、Web和地图与白宫取得联系。

为了开始该项目，先执行如下步骤，以在Eclipse中创建一个新的项目。

1. 单击File->New->Android Project，打开New Android Application向导。
2. 在Application Name文本框输入Leader。这也会自动在Project Name文本框输入Leader。
3. 在Package Name文本框中输入org.cadenhead.android。
4. 在Minimum Required SDK和Target SDK下拉列表中选择API 19，然后单击Next，以看到更多项目选项。
5. 确保在工作区中有3个复选框被选中：Create Custom Launcher Icon、Create Activity和 Create Project。
6. 单击Next，看到与应用程序图标相关的问题。
7. 单击Next，接受所有的默认设置。

8. 选择Fullscreen Activity，然后单击Next。
9. 在Activity Name文本框中输入LeaderActivity。
10. 单击Finish。

和SalutonMondo项目一样，该项目也会出现在Eclipse Package Explorer中。为了避免混淆，在进行下一步操作之前关闭SalutonMondo项目。在Package Explorer中右键单击SalutonMondo，然后从弹出菜单中选择Close Project。

Did you Know?

提示

该项目包含了大量的知识。随着工作的进行，你会发现，将浏览器打开并定位到Android Developer站点的Reference内容（<http://developer.android.com/reference>），将会为你提供很多帮助。你可以在Android类库中搜索Java类，以及项目中出现的文件的名字，以获悉更多内容。

24.4.1 组织资源

在创建Android app时，需要使用Java来编程，但是大部分工作是在Eclipse界面中完成的。在你完全精通Android SDK的功能时，不用编写一行Java代码就可以完整大部分工作。

为了无需编程，首先要创建供app使用的资源。每一个新的Android项目在开始时都有几个放置了资源的文件夹。要查看这些文件夹，在

Package Explorer中展开Leader文件夹，然后展开/res文件夹以及它所有的子文件夹（见图24.9）。

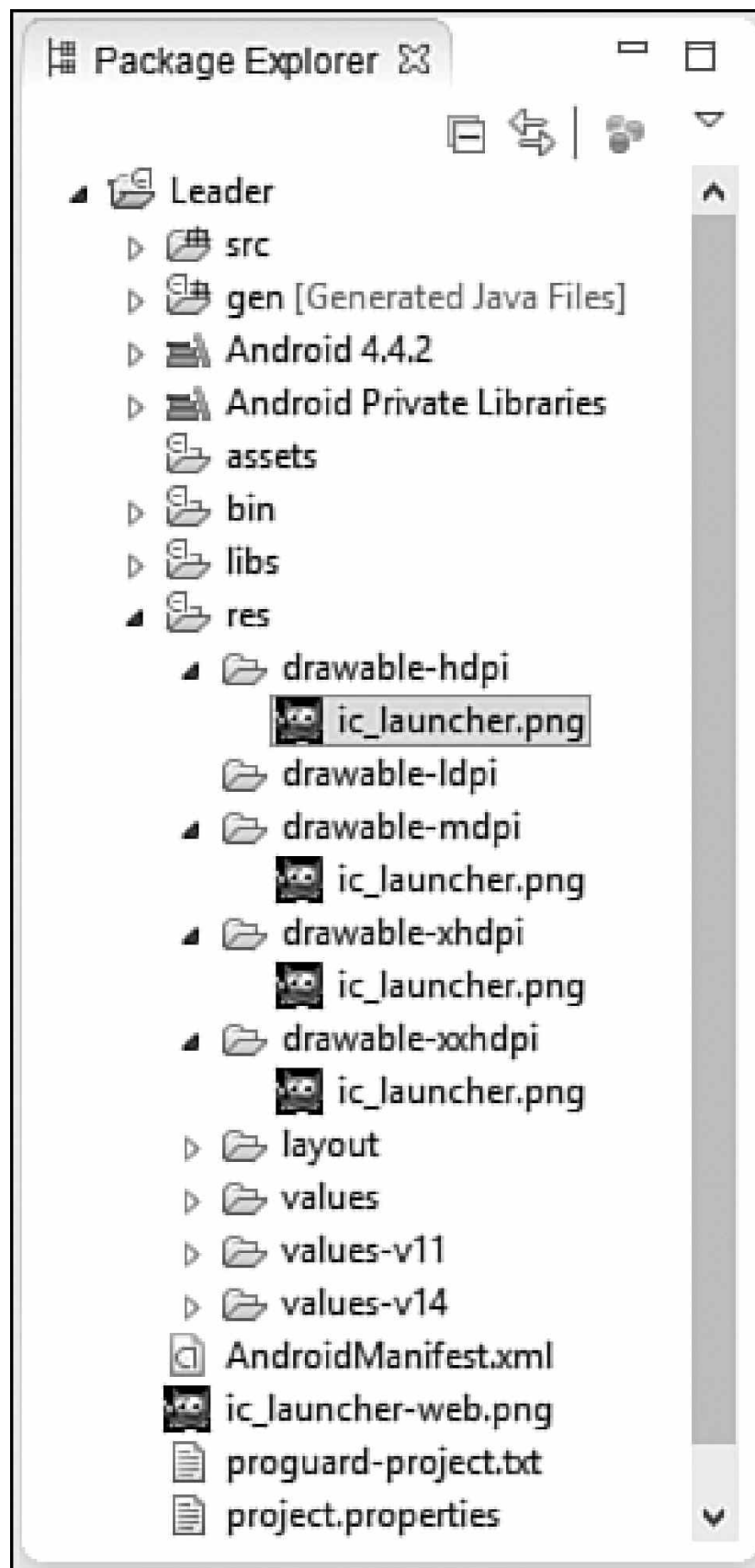


图24.9 查看app的资源文件夹

资源包含PNG/JPG/GIF格式的图形、存储在strings.xml文件中的字符串、XML格式的用户界面布局文件，以及你可以创建的其他文件。通常需要添加到项目中的两个文件是color.xml和styles.xml，其中前者是app中用到的颜色，后者定义了用户界面组件的外观。

新项目中的/res文件夹包含如下几个文件夹：drawable-hdpi、drawable-ldpi、drawable-mdpi、drawable-xhdpi和drawable-xxhdpi。其中每个文件夹中存放着ic_launcher.png的一个版本。ic_launcher.png是app的图标，这是一个比较小的图形，用来启用app。

ic_launcher.png的这些版本是相同的图形，但是分辨率不同。你在这里不会用到这些图标，因此可以将它们删除：在Package Explorer中单击一个ic_launcher.png文件，然后按下键盘上的Delete键。在删除每一个文件时，系统都会要求用户进行确认。

删除这些文件时，Package Explorer中将出现两个红色的X：一个位于AndroidManifest.xml上面，另一个位于最顶层的Leader上面（图24.10中对其进行了标识）。这些X符号标识该app发生了错误，该错误将阻止app的编译和运行。

由于app现在缺乏图标，因此引发了错误。在项目中添加一个新的图形文件appicon.png，然后在AndroidManifest.xml文件（app的主配置文件）中将其指定为该app的图标。

本书配套站点包含该app需要的appicon.png和另外4个图形文件：browser.png、maps.png、phone.png和whitehouse.png。访问www.java24hours.com，然后定位到第24章的页面，下载这5个文件，并将它们存放到计算机的临时文件夹中。

Android支持多个分辨率，但是这里不会用到。这里不是使用已有的drawable文件夹，而是创建一个新的文件夹，步骤如下。

1. 在Package Explorer中单击/res文件夹，将其选中。
2. 选择File->New->Folder，打开New Folder对话框。
3. 在Folder Name文本框中输入drawable。
4. 单击Finish。

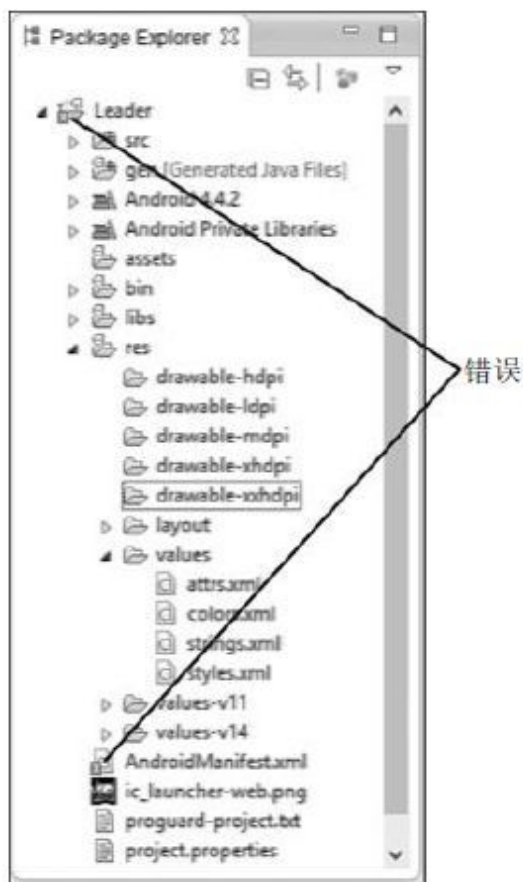


图24.10 检测和修复app中的错误

/res文件夹中将创建一个名为**drawable**的新文件夹。app需要的所有图形可以存到这里，而且不用考虑它们的分辨率。

使用拖放技术可以将文件添加到资源中。打开包含5个文件的临时文件夹，选择它们，然后拖放到**Package Explorer**中的**drawable**文件夹中（系统会咨询用户是复制文件还是复制文件的连接。这里选择**Copy Files**选项）。

现在该项目有了新的图标，可以将其设置为app的图标（通过编辑**AndroidManifest.xml**来实现），这样**Package Explorer**中出现的错误图标

将消失。

Watch Out!

警告:

app中的资源使用ID来表示，将资源名字的扩展名去掉后，就是其ID。比如，appicon.png的ID是appicon，browser.png的ID是browser。不同资源的ID不可能相同（以不同分辨率存储在drawable-*dpi文件夹中的同一个图形是一个例外，因为它们被当做一个资源）。

如果两个资源的名字相同，但是扩展名不同，比如appicon.png和appicon.gif，则Eclipse会标识错误，而且该app无法编译。

资源的名字只能包含小写字母、数字、下划线和点号。该项目中的文件遵循了这些规则。

24.4.2 配置app的Manifest文件

在Android app中，主要的配置工具是名为AndroidManifest.xml的文件，它位于app主文件夹中。app使用的所有XML文件都可以手动编辑，或使用Eclipse中内置的编辑器来编辑。其中后者使用起来比较方便，而且不容易发生错误。当你编写Android程序的经验逐渐丰富时，可以考虑修改XML文件，否则不建议修改。

为了为app选择适合的图标，请执行如下步骤。

1. 在Package Explorer中双击AndroidManifest.xml，该文件将在Eclipse内置的编辑器中打开。

2. 编辑器的底部位置将出现几个选项卡。单击**Application**选项卡来查看与该app相关的设置（见图24.11）。

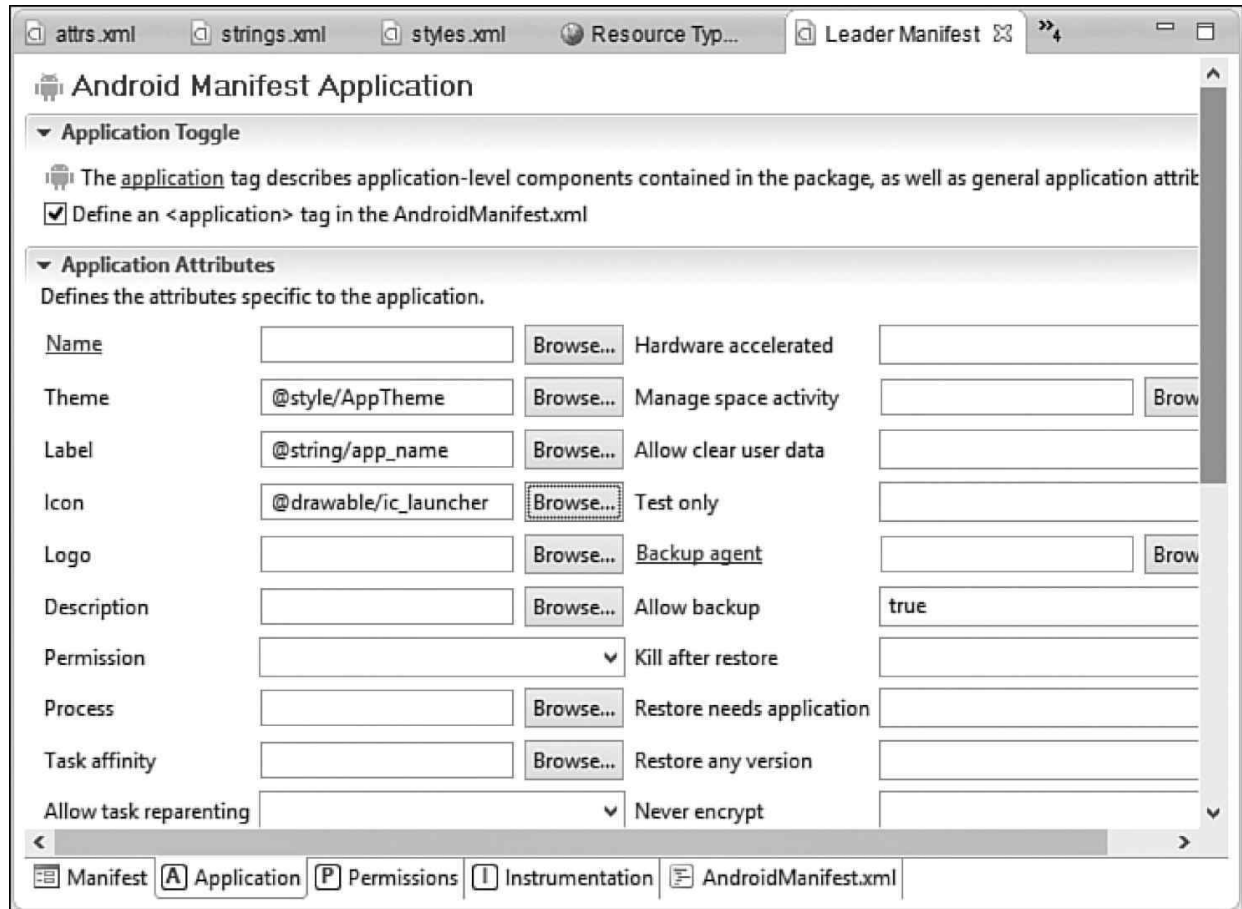


图24.11 编辑app的AndroidManifest.xml文件

3. **Icon**文本框表示app的图标，当前在文本框内的值 `@drawable/ic_launcher` 不正确。单击该文本框后面的**Browse**按钮，打开 **Resource Chooser**对话框，其中列出了你最近添加到app中的5个图形资源。

4. 选择appicon并单击**OK**。**Icon**文本框现在有了正确的值。

5. 保存文件：单击Eclipse工具栏中的Save按钮，或者是选择File->Save。

红色的X将从Package Explorer中消失，这表示已经为app指定了合适的图标。

24.4.3 设计用户界面

app的图形用户界面由布局组成，布局是用来放置文本框、按钮、图形和自定义控件的容器。向用户显示的每一个画面都有一个或多个布局。有的布局可以水平或垂直地叠放组件，有的布局可以将组件组织到表中，有的布局还有其他布置方式。

app可以有一个或多个画面。一个游戏可以包含如下画面：

- 游戏在载入时显示的运行画面；
- 带有按钮（用以查看其他画面）的主菜单画面；
- 显示游戏规则的帮助画面；
- 列出最高游戏得分的分数画面；
- 包含游戏开发人员的致谢画面；
- 游戏进行中的画面。

Leader app包含了一个画面，上面放置的按钮用来练习美国总统，或者其他领导人。

app的所有画面都存放在/res/layout文件夹中。该文件夹中存放了一个activity_leader.xml文件，该文件被指定为app在载入时所要显示的画面。

为了编辑该画面的布局，在Package Explorer中双击activity_leader.xml文件，将会在Eclipse主窗口中打开该画面，如图24.12所示。

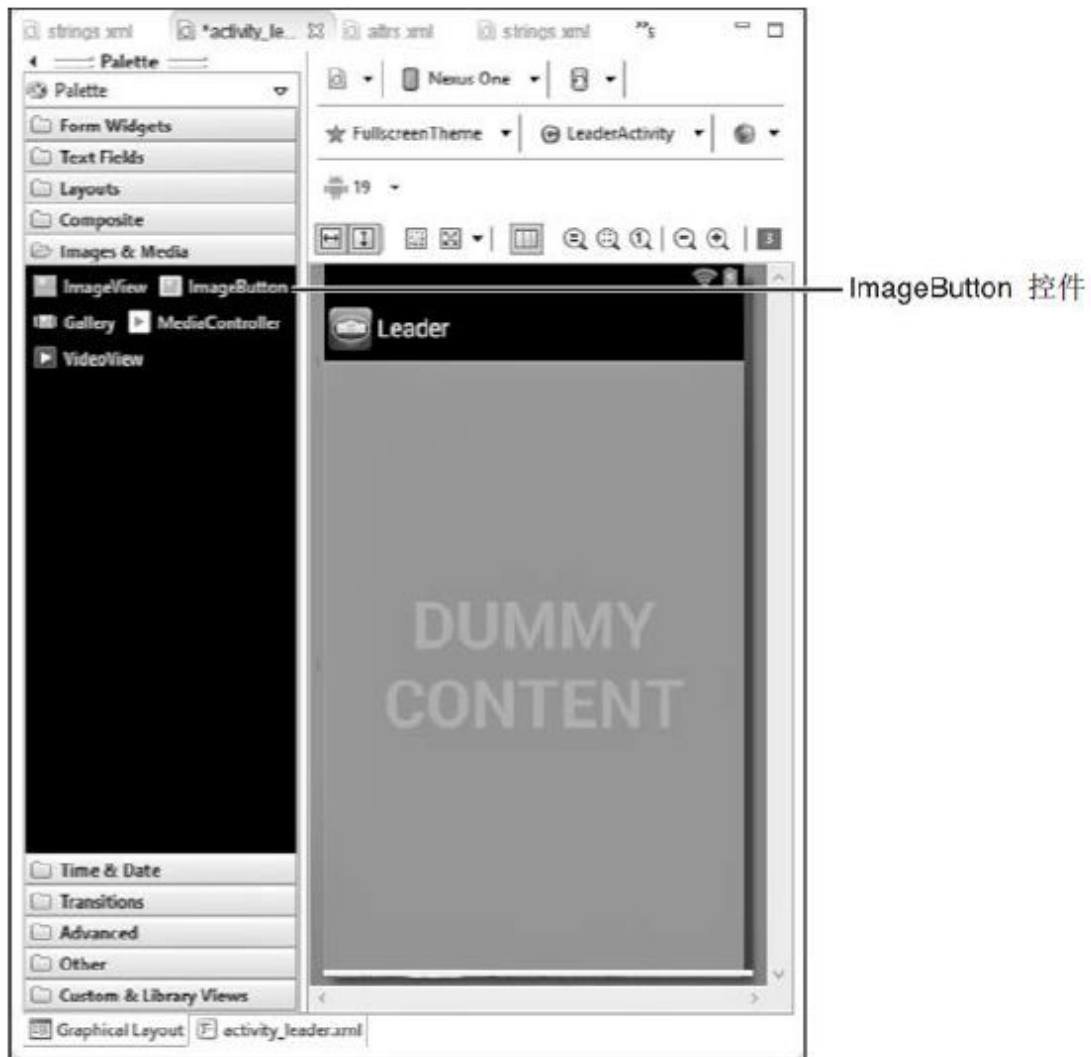


图24.12 编辑一个Activity的图形用户界面

编辑窗口中包含一个Palette面板以及几个可以展开的文件夹。Form Widgets子面板（有可能已经展开）显示一些简单的控件（widget），这些控件可以立刻被拖放到画面中。

按照如下步骤将3个图形按钮添加到画面中。

1. 删除显示“Hello World”文本的textview控件。在画面中单击该控件，然后按下Delete键。
2. 双击Palette面板中的Layouts文件夹，将其展开。
3. 从Palette中拖动LinearLayout（vertical）控件到画面中。画面中出现两个新的按钮，其中一个已经被选中：Set Horizontal Orientation。将布局拖放到画面中可以确定如何放置画面中的用户界面元素。
4. 单击Palette面板中的Images & Media文件夹，将其展开。
5. 从Palette中拖动一个ImageButton控件到画面中。打开一个Resource Chooser按钮，让用户选择要在按钮上显示的图像。选择phone然后单击OK。现在按钮上面有一个电话的图像。
6. 拖动另外一个ImageButton控件到电话按钮的右边。为该按钮选择browser，从而添加了一个浏览器按钮。
7. 在拖动一个ImageButton控件到浏览器按钮的右边，为其选择maps图像。
8. 一个Outline面板列出画面中的控件。从中选择imageButton1，该按钮的属性在Properties面板中打开（见图24.13）。

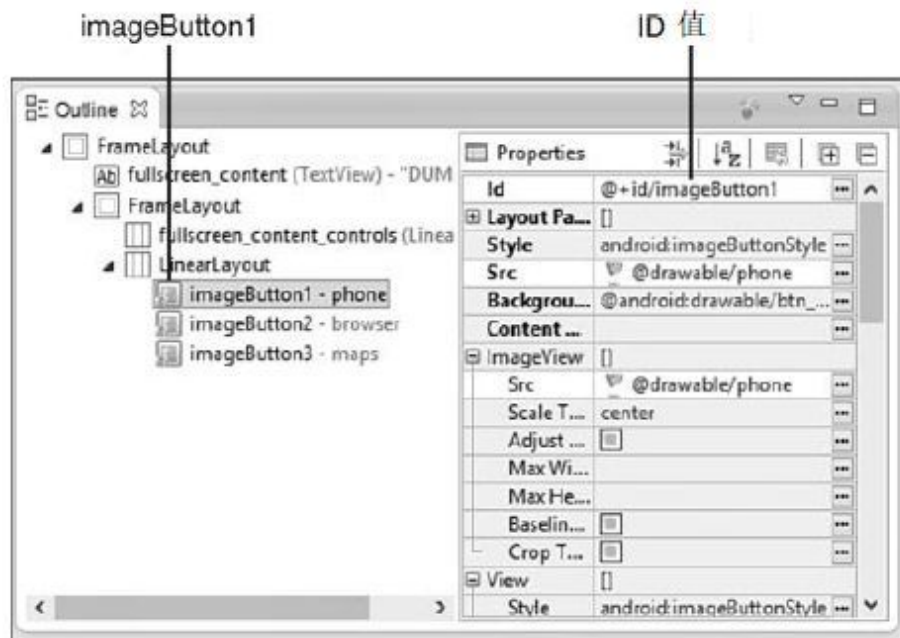


图24.13 自定义控件的属性

9. 滚动Properties面板，直到发现ID属性为止。ID属性的值当前被设置为@+id/imageButton1，将其修改为@+id/phonebutton。系统会提示用户是否更新所有引用以反映这一变化，为此可单击OK。

10. 在On Click属性中，输入值processClicks（将在下一节讲解）。

11. 为imageButton2重复步骤8~10，将其ID值修改为@+id/webbutton，On Click属性为processClicks。

12. 为imageButton3重复步骤8~10，将其ID值修改为@+id/mapbutton，On Click属性为processClicks。

13. 单击Outline面板中的LinearLayout，画面的属性将出现在Properties面板中。

14. 单击Background的值，然后单击椭圆形按钮 (...), 打开Reference Chooser。

15. 展开Drawable，选择whitehouse，然后单击OK。此时白宫的图片将成为画面的背景。

16. 单击Save按钮。

完成后的画面如图24.14所示。

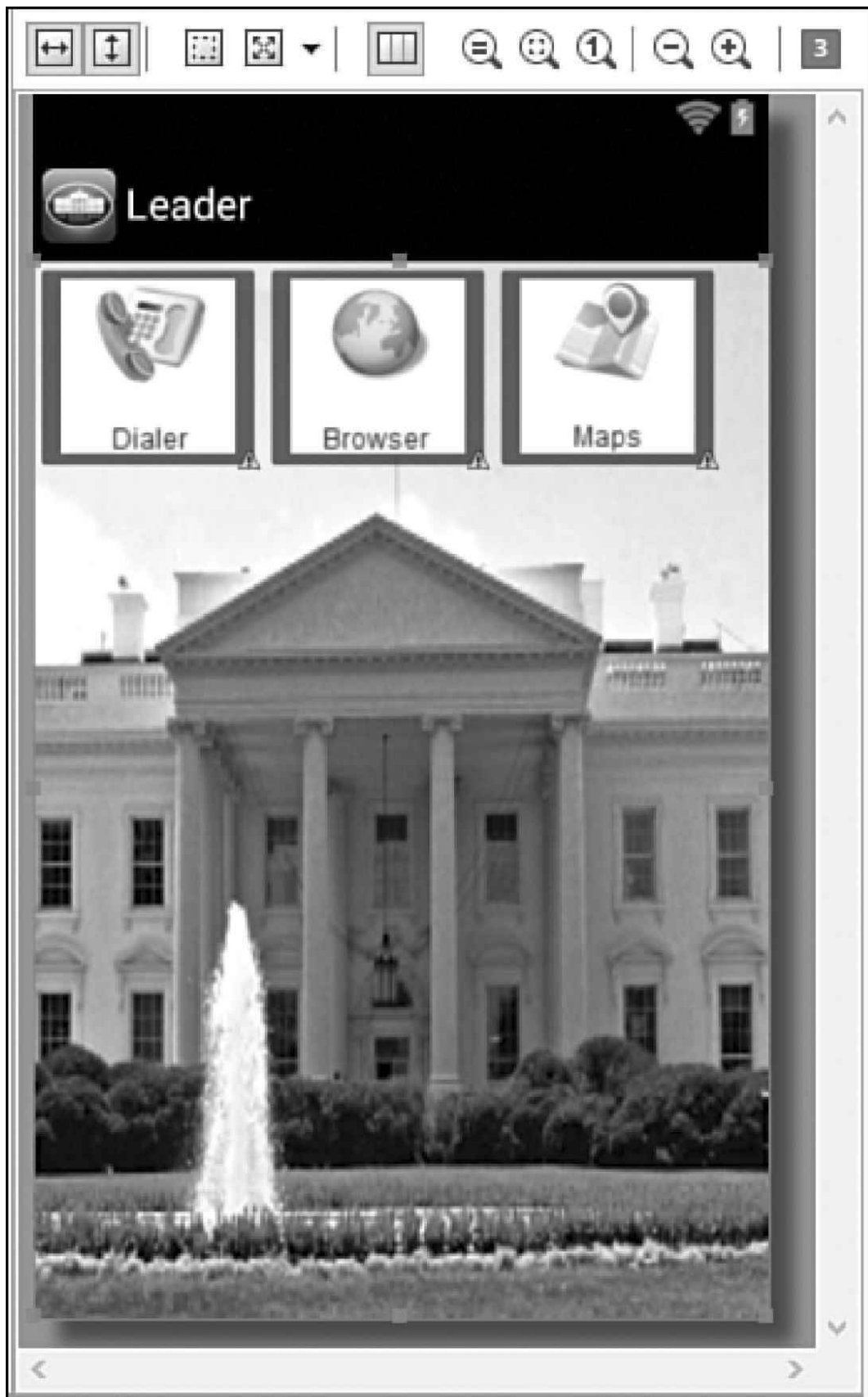


图24.14 预览app的图形用户界面

24.4.4 编写Java代码

此时，你已经完成了该app的大部分工作，但是你还没有编写一行Java代码。当你充分使用Android SDK的功能，而不求助于编程时，app的开发工作无疑会轻松很多。

app被组织为Activities，这表示app可以处理的任务。每一个Activity都由它所属的Java类来定义。当创建app时，需要确定名为LeaderActivity的Activity已经创建。这样，当app运行时，匹配该名字类能够自动运行。

LeaderActivity.java的源代码可以在Package Explorer中找到，它位于/src/org.cadenhead.android文件夹中。双击该文件，以进行编辑。

刚开始时，该类已经有了大量代码。

与所有Activity一样，LeaderActivity类是android.app包中Activity的子类，它包含了所需的行为来显示画面、收集用户输入、保存用户设置等。

当载入类时，类中定义的onCreate()方法将被调用。

该方法做的第一件事情是使用super()调用其超类中的同一方法：

```
super.onCreate(savedInstanceState);
```

接下来调用`setContentView()`，这个方法用来选择要显示的画面。该方法的参数是一个示例变量：`R.layout.activity_leader`，它会引用`/res/layout`中的`activity_leader.xml`文件。你可能还记得，资源的ID就是不带扩展名的文件名称。

在前面设计app的用户界面时，每个按钮的On Click属性被设置为`processClicks`，这表明当用户点击了画面中的一个按钮控件时，将调用`processClicks()`方法。

现在我们来实现这个方法。将下面的代码添加到`onCreate()`方法上面的`LeaderActivity`中：

```
public void processClicks(View display) {  
    Intent action;  
    int id = display.getId();  
}
```

该方法只有一个参数：来自`android.view`包中的`View`对象。`View`是app中的一些可视化显示。在本例中，`View`是包含拨号盘、浏览器和Maps按钮的画面。

`View`对象的`getID()`方法被单击按钮（`phonebutton`、`webbutton`或`mapbuttons`）的ID。

该ID存储在`id`变量中，以便在发生单击时可以在`switch`语句中执行某些操作：

```
switch (id) {  
    case (R.id.phonebutton):  
        // ...  
        break;  
    case (R.id.webbutton):  
        // ...  
        break;  
    case (R.id.mapbutton):  
        // ...  
        break;  
    default:  
        break;  
}
```

这段代码使用每个ID作为switch语句中的条件，执行3种操作中的一种。

方法processClicks()中的第一条语句创建一个变量来存放Intent对象。Intent是Android android.content包中的一个类：

```
Intent action;
```

在Android中，Intent用来实现Activity之间的通信。它也是app与Android设备的通信方式。

在这个方法中使用了3个Intent。

```
action = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:202-456-1111"));
```

```
action = new Intent(Intent.ACTION_VIEW, Uri.parse("http://  
➔ whitehouse.gov"));

action = new Intent(Intent.ACTION_VIEW, Uri.parse("geo:0,0?q=White  
House,  
➔ Washington, DC"));
```

Intent()构造函数接收两个参数:

要采取的操作, 由其类变量来表示;

与该操作相关联的数据。

这3个**Intent**分别告诉Android设备: 设置一个打给白宫的拨出电话, 其电话号码为 (202) 456-1111; 浏览<http://whitehouse.gov>网站; 使用部分地址“White House, Washington, DC”来载入Google地图。

在创建**Intent**之后, 下面的语句将让它执行相应的任务:

```
startActivity(action);
```

因为你已经删除了Activity上的Dummy Button, 因此必须移除这行代码, 它位于onCreate()方法的组后一行。

```
findViewById(R.id.dummy_button).setOnTouchListener  
➔ (mDelayHideTouchListener);
```

移除这行代码的最简单的方法是在它的前面放置//字符，将其变为可被编译器忽略的注释。

要让你添加的代码能正确运行，你必须将下面的import语句添加到类中：

```
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
```

LeaderActivity类中processClicks()方法的完整代码如程序清单24.1所示。确保你的代码与这里的完全一致。

程序清单24.1 LeaderActivity.java的完整源代码

```
1:      public void processClicks(View display) {
2:          Intent action;
3:          int id = display.getId();
4:          switch (id) {
5:              case (R.id.phonebutton):
6:                  Log.i(TAG, "Making call");
7:                  action = new Intent(Intent.ACTION_DIAL,
8:                      Uri.parse("tel:202-456-1111"));
9:                  startActivity(action);
10:                 break;
11:             case (R.id.webbutton):
12:                 Log.i(TAG, "Loading browser");
13:                 action = new Intent(Intent.ACTION_VIEW,
14:                     Uri.parse("http://whitehouse.gov"));
15:                 startActivity(action);
16:                 break;
```

```
17:         case (R.id.mapbutton):
18:             Log.i(TAG, "Loading map");
19:             action = new Intent(Intent.ACTION_VIEW,
20:                                 Uri.parse("geo:0,0?q=White House,
Washington, DC"));
21:             startActivity(action);
22:             break;
23:         default:
24:             break;
25:     }
26: }
```

输入完毕后，保存文件。它应该能够成功编译（Eclipse会自动执行）。如果没有，则那个熟悉的红色X会出现在Package Explorer中，用来指出在项目的哪个文件中发现了错误。

在排除了错误之后，就可以准备运行该app了。

首选，你必须为这个项目创建一个新的调试配置。

1. 在Eclipse的主工具栏中单击靠近Debug按钮的箭头，然后选择Debug Configurations，打开Debug Configuration对话框。

2. 双击左侧面板的Android Application，创建一个名为New_configuration（1）的新配置。

3. 在Name文本框中输入LeaderDebug。

4. 单击Browse按钮，选择Leader项目，然后单击OK。

5. 单击Target选项卡。

6. 在Deployment Target Selection Mode下选中Automatic，然后选择SimpleAVD Android虚拟设备。

7. 将Deployment Target Selection Mode更改为Always Prompt to Pick Device，单击Apply，然后单击Close。

现在，创建了一个名为LeaderDebug的新调试配置。

要运行该app，单击靠近Debug按钮的箭头，然后选择LeaderDebug（如果有的话）。如果没有，则选择Debug Configuration，再选择LeaderDebug，然后单击Debug。此时打开Android Device Chooser。选择Launch a New Android Virtual Device，然后选择SimpleAVD，然后单击OK。

在接下来的几分钟时间里，模拟器将载入并自动运行该app。

模拟器并不能模拟Android设备的一切行为。这个Leader app的Dialer和Browser按钮应该能够正常工作，但是在使用Maps按钮时，可能会遇到问题。

该app也可以在Android手机上运行，前提是该手机可以运行在Android SDK中，而且已经被设置为调试模式。

单击Debug按钮附近的箭头，然后选择LeaderDebug，此时该选项应该绝对存在。选择Choose a Running Android Device，在列表中选择你的手机，然后单击OK。

图24.15为运行在我手机上的app。当手机从肖像模式切换到风景模式时，该app也随之切换（该截图还显示我有141个新的语音信息，我应该查看一下）。

这个Leader app也将它自己的“Take Me to Your Leader”图标添加到手机应用程序中。即使你拔掉USB线缆，该app也会在手机上继续运行。

祝贺你！现在世界上有了10亿零1个Android app了。

By the Way

注意

读者可能也猜到，Android编程所需要的知识肯定要比本章讲解的内容要多。Sams出版社还出版了其他Android编程相关的图书，如《Sams Teach Yourself Android Application Development, 3rd Edition》（由Lauren Darcey和Shane Conder编写）。

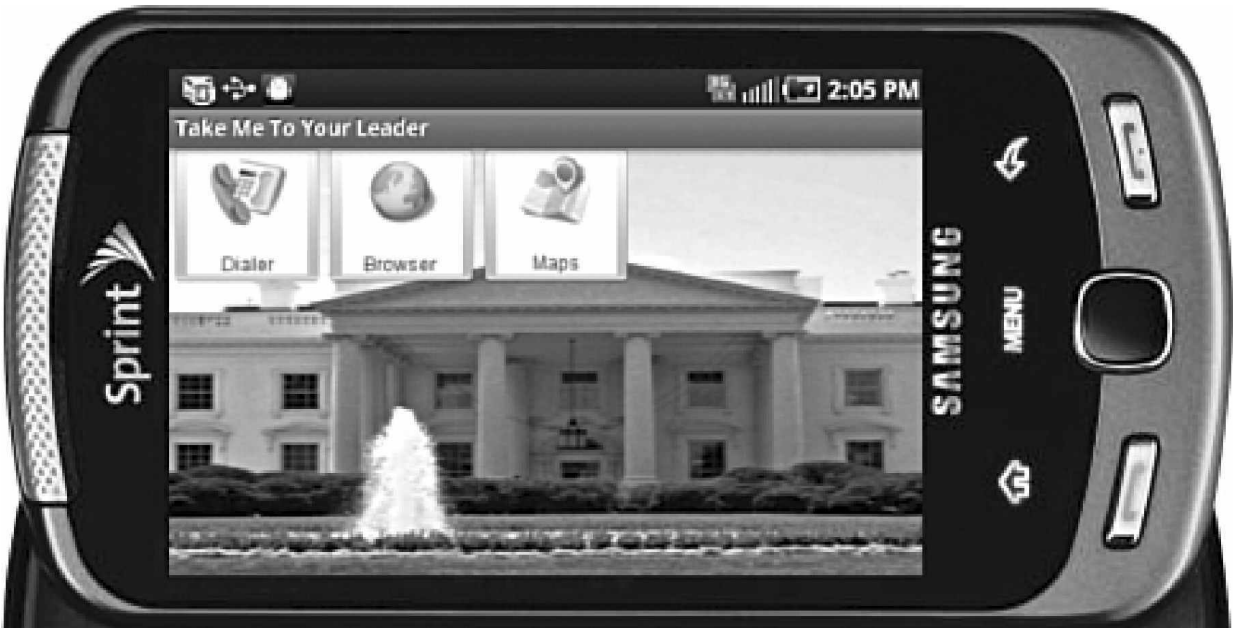


图24.15 在手机上运行该Leader app

24.5 总结

本书的目标是帮助读者熟悉编程的概念，提升编写应用程序的信心，无论应用程序是用于台式计算机、Web页面、Web服务器，还是手机。

当你具有了Java编程经验之后，其他相关的经验也会随之增长，因为像面向对象编程、虚拟机和安全环境等概念在软件开发中都处于前沿地位。

如果你的Java编程经验还有所欠缺，可以查看该书的附录，以寻找有用的信息。

在本章内容结束后，你可以通过多种途径继续学习Java语言。你可以访问[://weblogs.java.net](http://weblogs.java.net)站点来查看有关Java语言的讨论，在

www.careerbuilder.com这样的网站上会列出大量的与Java 编程有关的工作机会。此外，在该书的配套站点www.java24hours.com 上可以找到作者的信箱、查看该书练习的答案，以及勘误等内容。

作为一名Java程序员，你可以通过阅读*Sams Teach Yourself Java in 21 Days* 来进一步提到你的编程技能。我也是该书的作者，该书对这里讲到的各个主题进行了扩展，此外还添加了很多新的主题，比如JDBC、Java servlets和网络编程。本书还会涵盖Android的内容。

如果你是从头到尾阅读完了本书，期间没有略过什么内容，请用你新掌握的编程技能找到一个伟大的工作，然后重建世界经济吧。

24.6 问与答

问：为什么使用Eclipse而不是NetBeans来创建Android app？

答：也可以使用NetBeans来开发app，但是使用起来会相当麻烦，而且对Android编程的支持也不是很好。Eclipse被Google指定为首选的Android IDE。Android Developer站点（<http://developer.android.com>）上的官方文档和教程使用的都是Eclipse。

大多数的Android编程图书使用的也是Eclipse。尽管当你从NetBeans切换到Eclipse，以进行Android开发时，会存在一个适应阶段。但是当你掌握了app的编写、调试和部署基础之后，就会发现Eclipse更容易使用，因为它得到了Android程序员和技术作者更好的支持。

24.7 测验

如果你想测试一下刚刚学习到的Android开发知识，请回答下述问题。

24.7.1 问题

1. 下面哪一个公司不是开放手机联盟中的成员？

- a. Google。
- b. Apple。
- c. Motorola。

2. Activity之间使用什么对象进行通信？

- a. Intent。
- b. Action。
- c. View。

3. Android模拟器不能执行下面哪项任务？

- a. 接收短消息。
- b. 连接到Internet。
- c. 进行电话呼叫。

24.7.2 答案

1. b. Apple。Android是作为开源的一部分创建的，不涉及专利性，而且与Apple iPhone存在竞争关系。
2. a. Intent也是Activity与Android设备进行通信的方式。
3. c. 模拟器并不能模拟真实设备的所有行为，因此它只是测试app过程的一部分。

24.8 练习

为了进一步巩固你的Android知识，请做如下练习。

- 将SalutonMondo app的文本改为“Hello World”，然后在模拟器和Android设备（如果有的话）上运行该app。
- 针对不同的世界领导人创建Take Me To Your Leader app的新版本，并自定义电话、Web地址和地图信息。

有关为完成这些练习而编写的Java程序，请访问本书的配套网站 www.java24hours.com。

附录A 使用NetBeans IDE

尽管可以只使用Java开发工具包（JDK）和文本编辑器来开发Java程序，但是如果使用IDE，由此带来的编程体验会更好。

在本书的前23章，使用的是NetBeans，这是Oracle为Java程序员提供的一款免费IDE，它可以更容易地组织、编写、编译和测试使用Java开发的软件。NetBeans包含一个项目和文件管理器、图形用户界面设计器，以及许多其他工具。它的一个杀手级特性是用户在输入代码时，其代码编辑器能够自动检测到Java语法错误。

NetBeans的当前版本是8.0，它已经成为Java专业开发人员最喜欢的一款工具，而且NetBeans提供的功能以及自身的性能要物超所值。此外，它还是Java新手最容易上手的一款IDE。

在本附录中，你将学到如何安装NetBeans，以及如何将其用到本书中的项目中。

A.1 安装NetBeans

NetBeans IDE已经逐渐成为一款主流的Java编程工具。Java语言的发明人James Gosling在为图书*NetBeans Field Guide*作序时，提到“我一直使用NetBeans来开发Java程序”。当然，我也皈依到NetBeans门下。

NetBeans对3个版本的Java语言都提供支持，这3个版本是：Java Standard Edition（JSE）、Java Enterprise Edition（JEE）和Java Mobile Edition（JME）。它还支持Web应用开发、Web服务和JavaBeans。

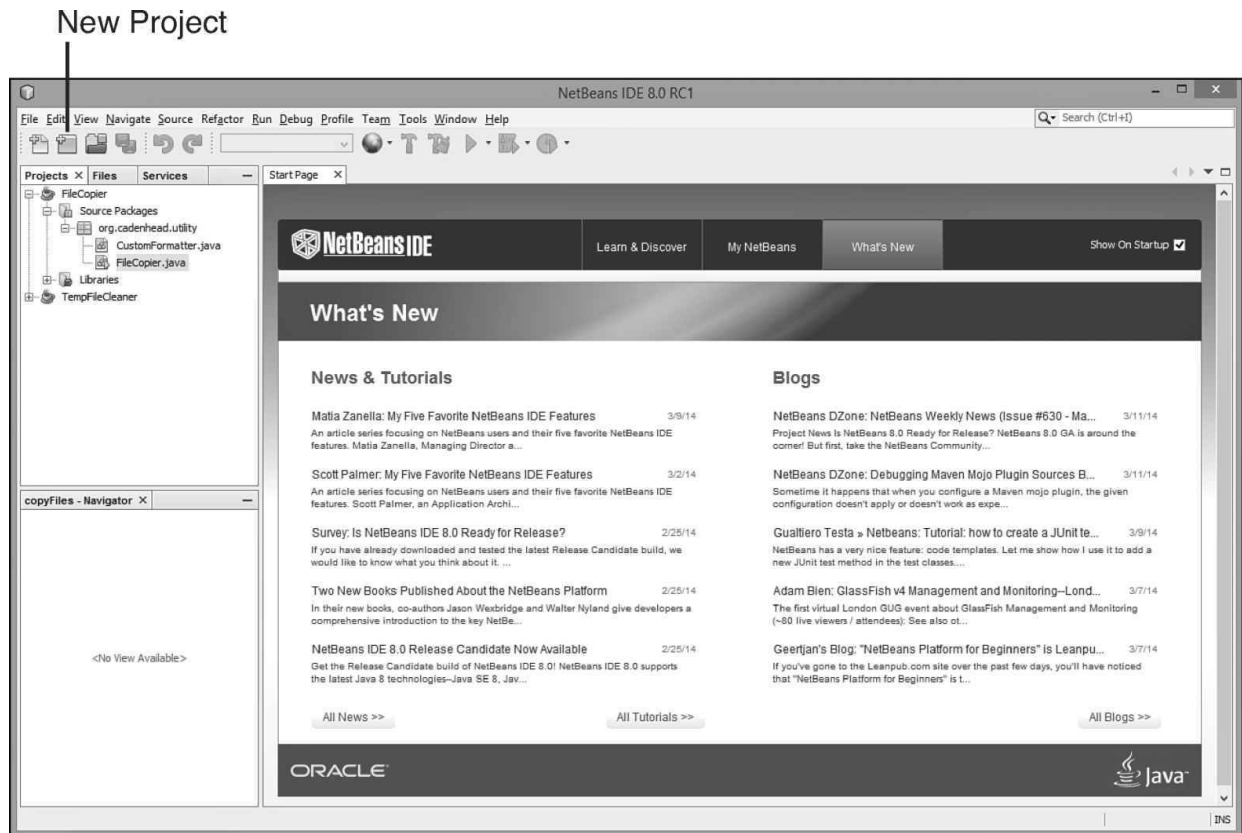
你可以从<http://netbeans.org>上下载该软件，它可以用于Windows、MacOS和Linux。下载NetBeans时，可以与Java开发工具包（JDK）一起下载，也可以单独下载。你必须在计算机上安装NetBeans和JDK。

如果你想确保下载的NetBeans和JDK的版本与本书中使用的相同，可以访问本书配套站点www.java24hours.com。单击该书的封面，在新打开的页面中即可看到下载JDK和NetBeans 8.0的链接。可以选择从此处下载。

A.2 创建新项目

下载JDK和NetBeans时，其方式与在计算机上安装软件相同，只不过下载时用到的是下载向导，而安装时用到的是安装向导。你可以根据自己的喜好，将该软件安装到任何文件夹和菜单组中，但是最好不要修改其默认的位置，除非你有很好的理由来这么做。

安装NetBeans后，当第一次运行它时，将会看到一个起始页，它显示相关新闻和编程指南的链接（见图A.1）。可以使用NetBeans内置的Web浏览器在该IDE中来阅读这些内容。



图A.1 NetBeans用户界面

NetBeans项目包含一组相关的Java类、这些类使用的文件，以及Java类库。每一个项目都有自己的文件夹，你可以使用文本编辑器和其他编程工具来探索和修改NetBeans的外观。

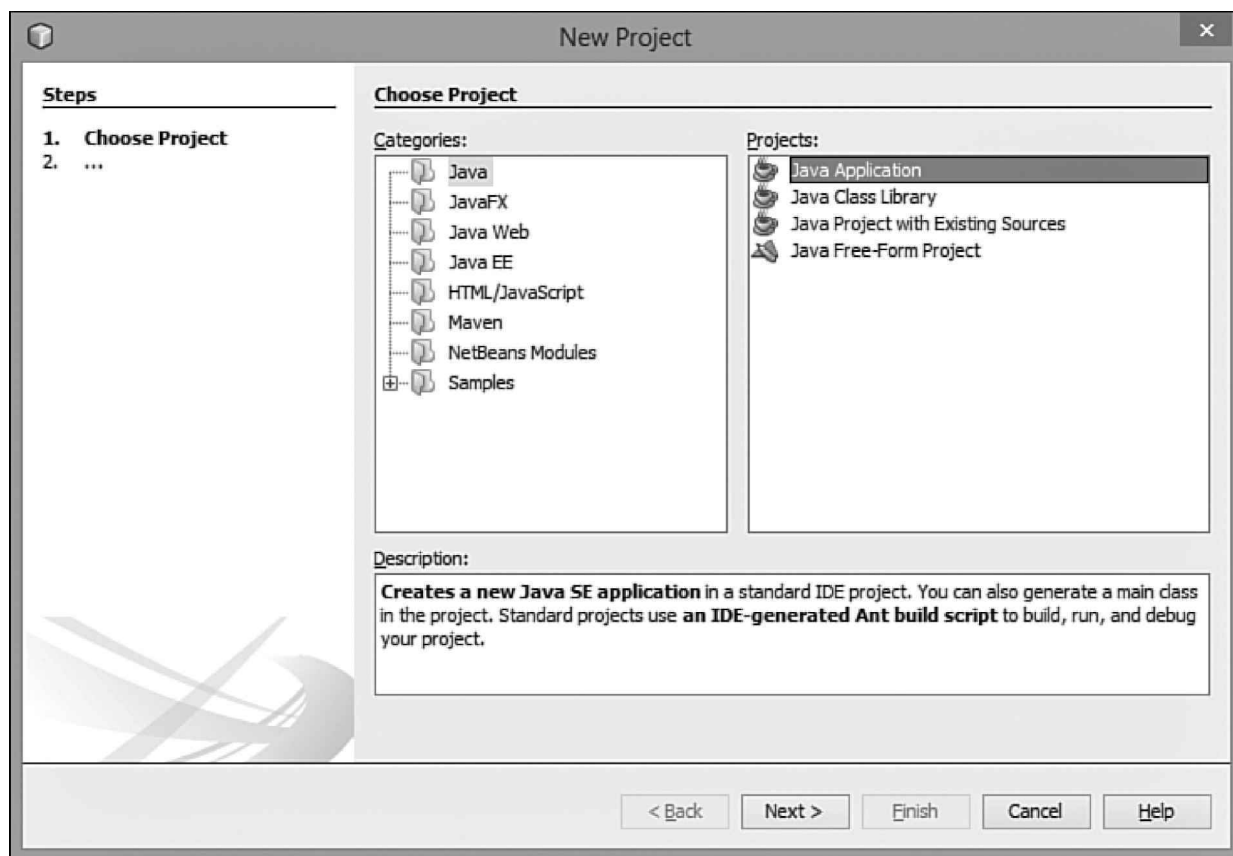
为了创建一个新的项目，单击图A.1中的New Project按钮，或者是选择File->New Project菜单命令。打开New Project Wizard，如图A.2所示。

NetBeans可以创建不同类型的Java项目，但是在本书中，你只需要关注Java Application即可。

对于你的第一个项目（以及本书中大多数项目）而言，选择项目类型**Java Application**，然后单击**Next**。该向导会让用户选择项目的名字和位置。

Project Location文本框指明了使用**NetBeans**创建的编程项目的根文件。在**Windows**中，这将是**My Documents**（或**Documents**）文件夹的子文件夹，名为**NetBeansProjects**。你创建的所有项目都存放在该文件夹中，而且每一个项目都有自己的子文件夹。

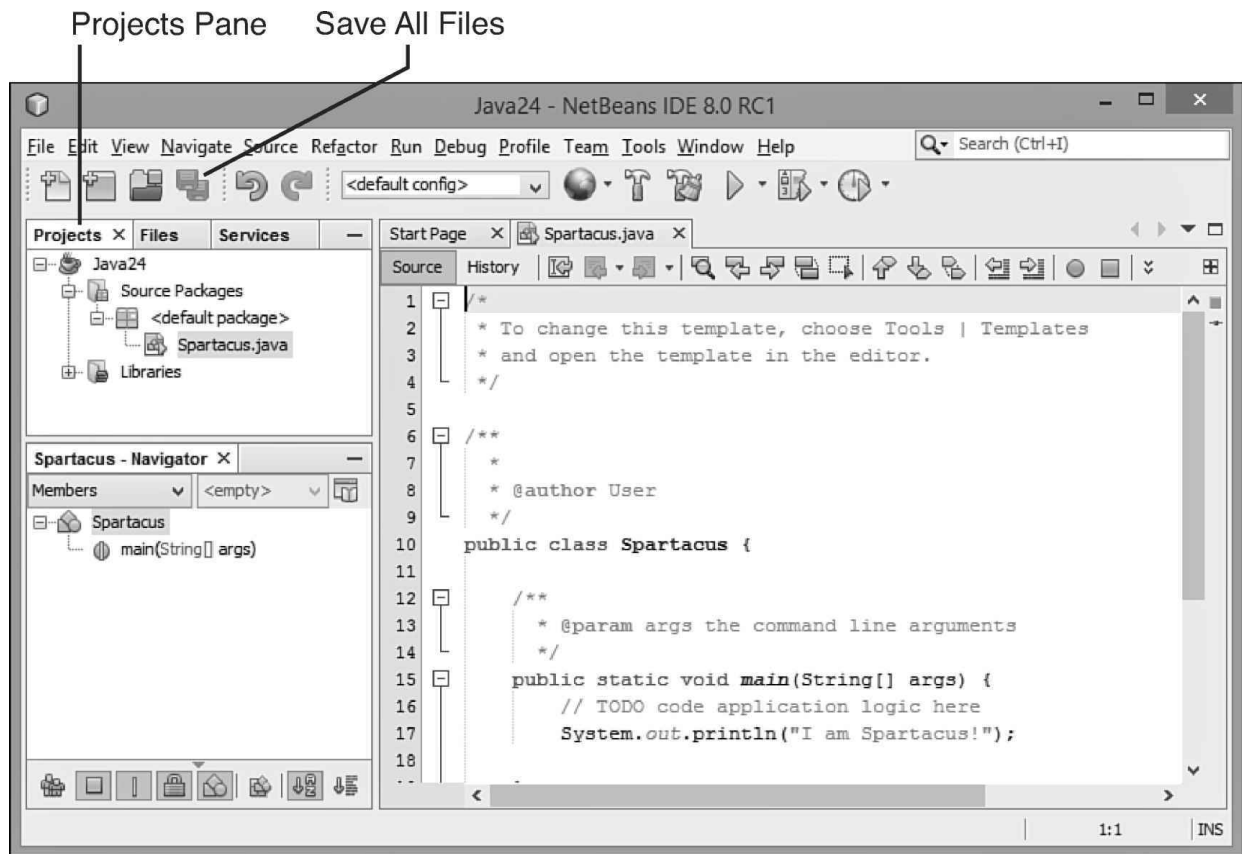
在**Project Name**文本框中输入**Java24**。**Create Main Class**文本框也随之改变，并推荐使用**java24.Java24**作为项目中**Java**主类的名字。将其修改为**Spartacus**，然后单击**Finish**，接受所有其他的默认值。现在**NetBeans**创建了项目和该项目的第一个类。



图A.2 New Project Wizard

A.3 创建新的Java类

当NetBeans创建了一个新的项目时，它将设置所有需要的文件和文件夹，然后创建主类。图A.3为项目Spartacus.java的第一个类，它在源代码编辑器中打开。



图A.3 NetBeans源代码编辑器

Spartacus.java是Java类的一个框架，只包含一个main()方法。类中以淡灰色显示的所有代码行是注释，其存在的目的是解释类的目的和功能。当类在运行时，会忽略掉注释。

要让类执行一定的任务，在注释行// TODO code application logic here下面添加一行新的代码：

```
System.out.println("I am Spartacus!");
```

`System.out.println()`方法显示一个文本字符串，在本例中是“I am Spartacus!”。

要确保你输入的内容和上面的一致。在确保输入的代码行没有错误之后，使用分号结尾，然后单击**Save All Files**工具栏按钮，来保存该类。

By the Way

注意

在输入时，源代码编辑器能识别你在做什么，并弹出有该**System**类相关的帮助信息，即**out**实例变量和**println()**方法。你以后会爱上这一点，但是现在先将其忽略。

在运行**Java**类之前，必须先将其编译为可执行的字节码。**NetBeans**会尝试自动编译类，也可以使用两种方式手动编译类：

- 选择菜单命令**Run->Compile File**。
- 在**Project**面板中右键单击**Spartacus.java**，在弹出的菜单中选择**Compile File**。

如果**NetBeans**不允许你选择这两种方式，则意味着它已经自动编译了类。

如果在编译类时失败，则**Project**面板中靠近文件名**Spartacus.java**的位置会出现一个红色的惊叹号。为了修复该错误，请将你在源代码编辑器中输入的内容与程序清单A.1中列出的源代码进行比较，如果没有

问题，再次保存。程序清单A.1中的行号不要出现在你的程序中——它们在本书中的目的是用来表述代码是如何工作的（同样，第8行会让你用自己的名字来替换单词“User”）。

程序清单A.1 Spartacus.java类

```
1: /*
2:  * To change this template, choose Tools | Templates
3:  * and open the template in the editor.
4:  */
5:
6: /**
7:  *
8:  * @author User
9:  */
10: public class Spartacus {
11:
12:     /**
13:      * @param args the command line arguments
14:      */
15:     public static void main(String[] args) {
16:         // TODO code application logic here
17:         System.out.println("I am Spartacus!");
18:
19:     }
20:
21: }
```

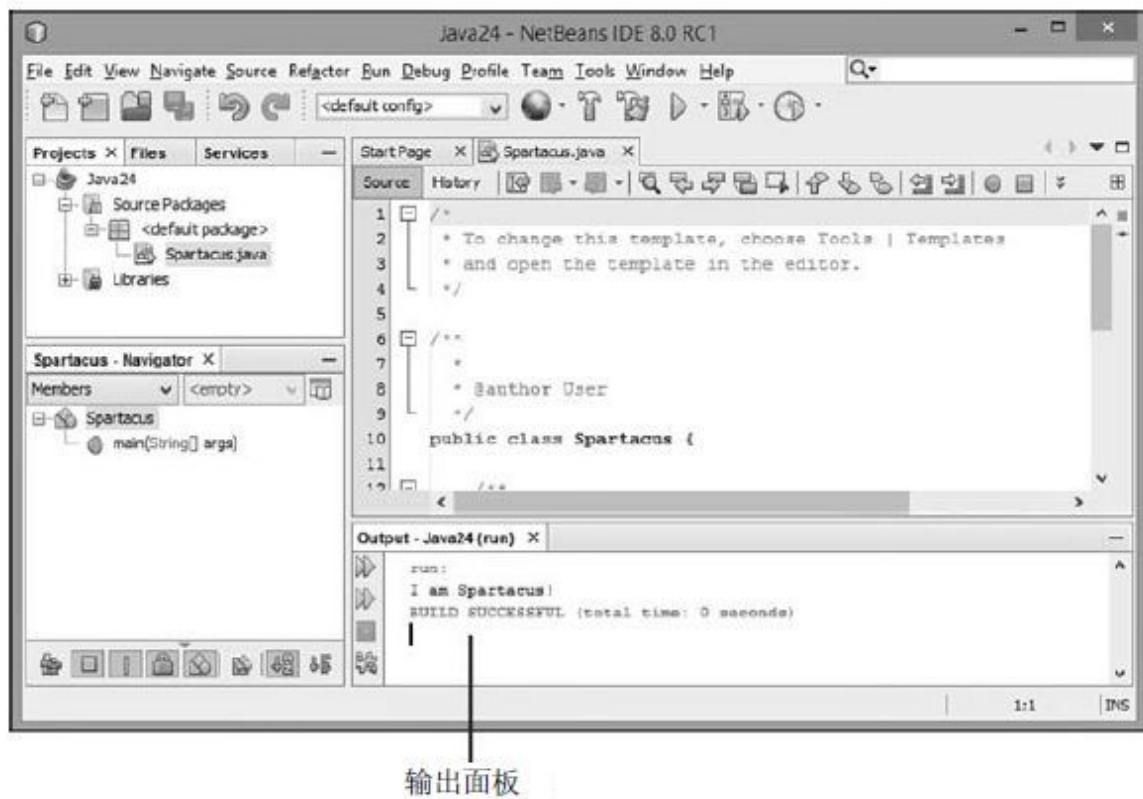
该类在第10～21行中定义。第1～9行是当你将项目类型选择为Java Application时，NetBeans在每一个新类中都会添加的注释。这些注释有助于解释与程序相关的信息，以方便人们阅读源代码。编译器会忽略这些注释。

A.4 运行应用程序

在创建并成功编译Spartacus.java类之后，可以使用两种方式在NetBeans内运行。

- 从菜单中选择 Run->Run File。
- 在Projects面板中右键单击Spartacus.java，然后在弹出的菜单中选择Run File。

当运行Java类时，其main()方法将被Java虚拟机调用。字符串“I am Spartacus!”将显示在输出面板中，如图A.4所示。



图A.4 Spartacus应用程序的输出

Java类要想运行，则必须有一个main()方法，如果试图运行一个不包含该方法的类，则NetBeans会报错。

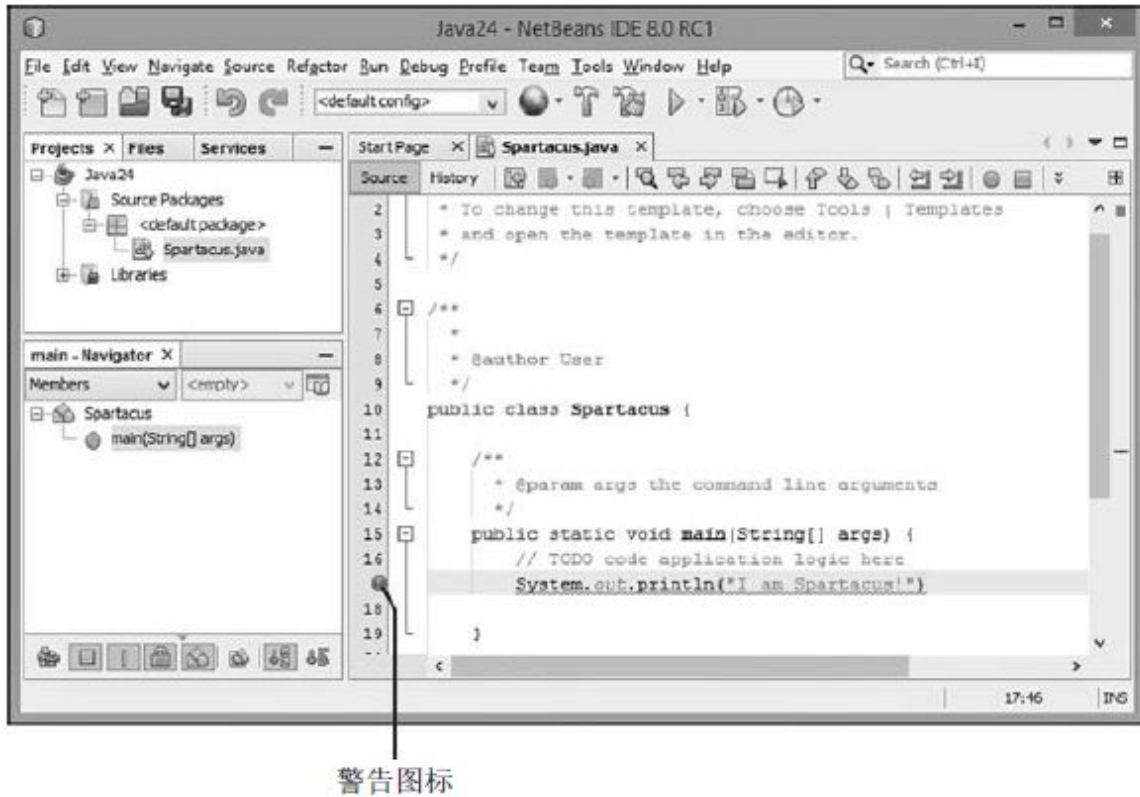
当查看完程序的输出后，单击面板选项卡上的X标记，关闭Output面板。这将增大源代码编辑的面积，当你在编写程序时，这会很方便。

A.5 修复错误

Spartacus应用程序已经编写、编译、运行完毕。现在，我们来看一下，当程序运行错误时，NetBeans会如何响应。

与其他程序员一样，你以后会通过大量的实践来提升自己的错误修复能力，但此时还是请多加注意。

返回源代码编辑器中的Spartacus.java，然后删除调用System.out.println()方法的那一行（程序清单A.1中的第17行）代码后面的分号。甚至在你保存该文件之前，NetBeans都会报错，并在相应行的左边显示一个红色的警告图标（见图A.5）。



图A.5 在源代码编辑器中标记错误

将鼠标放到该警告图标上，会出现一个对话框，该对话框描述了NetBeans发现的错误。

NetBeans源代码编辑器能够识别大多数常见的编程错误和输入错误，这些错误通常出现在编写Java程序时。它将阻止该文件被编译，直到错误被排除。

将分号放回代码原处后，则错误图标消失，此时可以保存并运行该类。

在创建和编译本书中的Java程序时，将会用到这些基本的特性。

除了这里提到的这些特性之外，**NetBeans**还有很多其他特性，但是在深入学习**NetBeans**之前，应该重点关注**Java**的学习。因此在刚开始使用**NetBeans**时，最好将其当成一个简单的项目管理和文本编辑器。然后利用它编写类、标记错误，以确保能成功编译和运行每一个项目。

当你准备深入学习**NetBeans**时，可以查看**NetBeans**的起始页，该起始页提供了很多资源来介绍如何使用**NetBeans**。Oracle也提供了培训和文档资源，地址为www.netbeans.org/kb。

附录B Java资源

结束本书的学习之后，你可能想知道如何进一步提高Java编程技能。本附录列出了一些图书、Web站点、Internet讨论组和其他资源，你可以通过它们来丰富Java知识。

B.1 可以考虑的其他图书

Sams出版社和其他出版社出版了一些与Java编程相关的图书，其中有些是对本书内容的进一步深入。这些图书如下所示。

- Sams Teach Yourself Java in 21 Days, by Rogers Cadenhead, ISBN 0-672-33710-x。尽管本书的前7章看似多余，但是它对Java进行了详细讲解，并添加了很多高级主题。如果你想使用另外504个小时来学习Java，则本书无疑很合适。
- The Java EE 6 Tutorial: Basic Concepts, Fourth Edition, by Eric Jendrock and others, ISBN 0-13708-185-5。该书讲解了Java Enterprise Edition (JEE) 的相关知识，JEE是Java类库的一个扩展，主要在大型计算环境中的大型企业中使用。
- Java Phrasebook, by Timothy R. Fisher. ISBN 0-67232-907-7。该书囊括了100多个Java代码案例，这些代码是由专业程序员和Java Developer's Journal杂志的供稿人开发的，你可以将其用在自己的Java程序中。

- **Agile Java Development with Spring, Hibernate and Eclipse** by Anil Hemrajani. ISBN 0-672-32896-8。该书以Java Enterprise Edition为讲解主体，向读者展示了如何使用Spring框架、Hibernate库和Eclipse IDE来降低其企业应用编程中的复杂度。
- **Android Programming Unleashed** by B.M. Harwani. ISBN 0-13-315175-1。以Android 4.0.3（冰淇淋三明治）和4.1（果冻豆）为主题的编程教程，涵盖了使用Java来编写Android app的相关基础、app设计的构建模块、app部署，以及其他高级主题，比如动画、使用谷歌地图、发送短消息和邮件。

读者可以从www.informit.com上免费下载摘自Java学习资源，这些资源来自于Sams出版社出版的其他Java图书。

读者从Sams出版社的站点www.informit.com/sams 可以找到Sams以及Pearson Technology Group旗下其他出版社将要出版的新书。

B.2 Oracle公司的Java官方网站

Oracle公司的Java软件部维护了3个网站，Java用户和Java程序员可能会对这些网站感兴趣。

查找有关Java的信息时，应首先访问网站hwww.oracle.com/technetwork/java。从这里可下载最新的JDK和其他编程资源以及完整的Java类库文档；另外还有bug数据库、用户组目录和支持论坛。

网站`www.java.net`是一个Java程序员大型社区。你可以编写与Java语言相关的博客，创建一个新的开源项目，并放到上面与大家共享，你还可以通过该站点与其他程序员进行合作。

网站`www.java.com`旨在让Java语言给消费者和非程序员带来更大好处。你可以从这个网站下载Java运行时环境，以便让用户在自己的计算机中运行使用Java语言开发的程序。此外，这里还有一个展览室，它通过Java示例来向用户显示Java在当今世界中的用途。

Java类文档

在Oracle公司的Java网站中，最有用的可能是关于Java类库中每个类、变量和方法的文档。

数千页的在线免费材料演示了如何在程序中使用这些类。

要查看Java 8的类文档，请访问
`http://download.java.net/jdk8/docs/api`。

B.3 其他Java站点

随着Java在Web页面上的应用，Java创造了巨大的奇迹，因此出现了大量专门介绍Java和Java编程的网站。

B.3.1 本书英文版的配套网站

本书英文版的官方网站为`www.java24hours.com`，附录C将对该网站做详细介绍。

B.3.2 Workbench

我经常通过博客Workbench讨论Java、Internet技术、计算机图书，以及其他相似的主题，其网址为<http://workbench.cadenhead.org>。

B.3.3 Slashdot

自从1997年以来，技术新闻网站Slashdot就成为程序员以及其他计算机行业从业人员的必去之地。该站点在其首页放置了用户提交的最佳故事，而且可以允许用户对评论进行打分，从而过滤掉噪音。要查看最新的Java故事，请访问www.slashdot.org/tag/java。

B.3.4 其他Java博客

还存在几百个与Java编程相关的博客，其中有些博客主要以Java编程为主，有些则不是。搜索引擎IceRocket在www.icerocket.com站点提供了与Java有关的最新博客列表。

WordPress.com托管了数千个博客，它将最新的Java相关的博文进行分类，并放在了<http://en.wordpress.com/tag/java>上。

B.3.5 InformIT

InformIT是一个技术参考网站，是Sams出版社支持的一个综合性网站，其网址为www.informit.com。该网站涵盖了十几个与软件开发和Internet相关的主题。InformIT的Java社区包括“How-to”文章和初学者指南。

B.3.6 Stack Overflow

在线社区Stack Overflow是一个程序员可以提供问题，并对其他用户的答案进行评价的地方。该网站为tagged类型，因此在搜索时，可以将搜索缩窄为感兴趣的语言或主题。要查看Java相关的问题，请访问<http://stackoverflow.com/question/tagged/java>。

B.3.7 JavaWorld杂志

自Java语言诞生之初，就有了该杂志，它经常发表教程性文章、Java进展新闻及其他专题，它还有视频和音频播客。它的网址为www.javaworld.com。

B.3.8 Developer.com's Java Directory

由于Java是一种面向对象语言，因此很容易在自己的程序中使用他人创建的资源。开发重要的Java项目前，应在网上查找可在程序中使用的资源。

一个不错的地方是Developer's Java Directory。该网站对Java程序、编程资源和其他信息进行分类，网址为www.developer.com/java。

B.3.9 Twitter

要在一个互动性更强的地方查看来自Java程序员的建议，可以试试Twitter。这是一个非常流行的微博服务，有数百万人使用它向朋友及其粉丝发送短消息。

通过使用#java标签可以识别与Java相关的信息，当然这可能也会涉及Java岛以及咖啡，这是因为这个标签是用户创建的，因此不正式。

要在Twitter上搜索与Java相关的最新消息，可以在浏览器中打开 <http://search.twitter.com>，然后搜索#java。

附录C 本书站点

阅读本书后，读者肯定有不太明白的地方，虽然作者不希望如此。

编程是一种专业性很强的技术，包含奇怪的概念和术语，如实例化、三元运算符以及高位优先字节序和低位优先字节序等。

如果读者对本书介绍的任何主题还有不清楚的地方，请访问本书英文版配套网站以寻求帮助，网址为www.java24hours.com（见图C.1）。

图C.1 本书的配套网站

该网站提供了以下内容。

- 勘误和说明：作者发现本书中的错误后，将在该网站上进行说明——提供正确的内容和其他有帮助的材料。
- 解答读者提出的问题：如果读者提出的问题没有包含在本书的“问与答”中，我将把它放到这个网站上。
- 本书所有程序所需的源代码、类文件和资源。
- 每章最后的练习答案，包括源代码。
- 本书提到的网站的最新地址：如果本书提到的某个网站的地址发生变化，且我知道这个新的URL，我会将其放在网上。

读者也可以通过访问本书英文版的配套网站来给作者发E-mail。单击**Feedback**链接将打开一个页面，在该页面中可以直接给作者发E-mail。

请读者随意发表看法，无论是积极的、消极的、冷淡的、模糊的、感兴趣的.....

——Rogers Cadenhead

附录D 设置Android开发环境

尽管Android app是使用Java语言开发的，但是在开发时仅使用标准的Java编程工具还不够，还需要Java开发工具包（JDK）、Android软件开发工具包（SDK），以及为Android编程量身打造的集成开发环境和Android设备的驱动程序。

Eclipse是用来开发Android app的最流行的IDE，而且对Android的支持最好。

在该附录中，你将会设置这些工具，并确保它们能够一起工作，以运行Android app。这些工具都是免费的，而且可以从Internet上下载。

D.1 起步

你可以在如下操作系统上进行Android编程：

- Windows XP或后续操作系统；
- Mac OS X 10.5.8或后续操作系统；
- Linux。

为了安装Android SDK，你需要有600MB的磁盘空间，而安装Eclipse IDE则需要1.2GB。

此时，你应该安装了Java开发工具包，因为本书中的程序都是使用Java开发工具包和NetBeans开发的。开发Android app需要JDK 6.0或更高版本。

如果出于某种原因，你仍然需要JDK，则可以从<http://jdk8.java.net/download> 上下载。

D.2 安装Eclipse

尽管其他IDE（比如NetBeans）也支持Android开发，但是Eclipse已经成为编写Android app的最常用的工具。Android开发人员将Eclipse作为他们首选的开发环境，而且他们的官方文档和教程中使用的也是Eclipse。

Eclipse与NetBeans一样，都提供了编写Java程序的图形用户界面。你可以使用它来创建任何类型的Java程序，而且它还支持其他编程语言。

Android需要的Eclipse版本为3.7.2或更高版本。使用Eclipse开发Android app时，也需要使用Android SDK。

By the Way

注意

比较常见的Android编程教程中使用的也是Eclipse，比如Sams Teach Yourself Android Application Development in 24 Hours, 3rd Edition（Carmen Delessio、Lauren Dercey和Shane Conder编写）。由于在本附录设置的工具

也可以用在上面提到的这本Android开发图书中，因此你可以在学习完本书后接着学习这本Android开发图书。

可以通过<http://developer.android.com/sdk>同时下载Eclipse和SDK。该站点提供了两个版本：ADT Bundle（Eclipse和SDK捆绑到一起）和SDK Tools Only（没有Eclipse）。为你的操作系统（Windows、Mac OS或Linux）选择ADT Bundle版本。

这个版本被打包为一个ZIP压缩文件，它包含一个eclipse文件夹、一个android文件夹和一个SDK管理器程序。在计算机中安装时，没有安装向导。文件夹eclipse中存放了运行Eclipse所需的所有文件。文件夹android中包含Android SDK，而SDK管理器程序用来管理SDK和安装新的版本。

在用来存储程序的文件夹中创建一个Eclipse子文件夹。在我的Windows系统中，我将它放在了Program Files（x86）文件夹中。解压eclipse和android文件夹以及SDK管理器，然后将它们放到eclipse文件夹中。

解压完毕之后，进入刚创建的eclipse文件夹，查找可执行的Eclipse应用程序。创建该应用程序的一个快捷方式，将其放到菜单中，或者是方便运行该程序的其他位置，比如桌面或任务栏中。

***By the
Way***

注意

在Windows中，要在安装程序的文件夹中创建一个快捷方式，可以右键单击Eclipse应用程序文件，然后选择Create Shortcut（创建快捷方式）。这会弹出一个对话框，询问是否要将快捷方式放到桌面上，单击Yes。你也可以将快捷方式拖到桌面上。

D.3 安装在Eclipse中使用的Android插件

Eclipse IDE支持多种编程语言和技术，但并不是立刻就可以支持的。只有在安装了相对应的插件之后，才能提供所需要的功能。

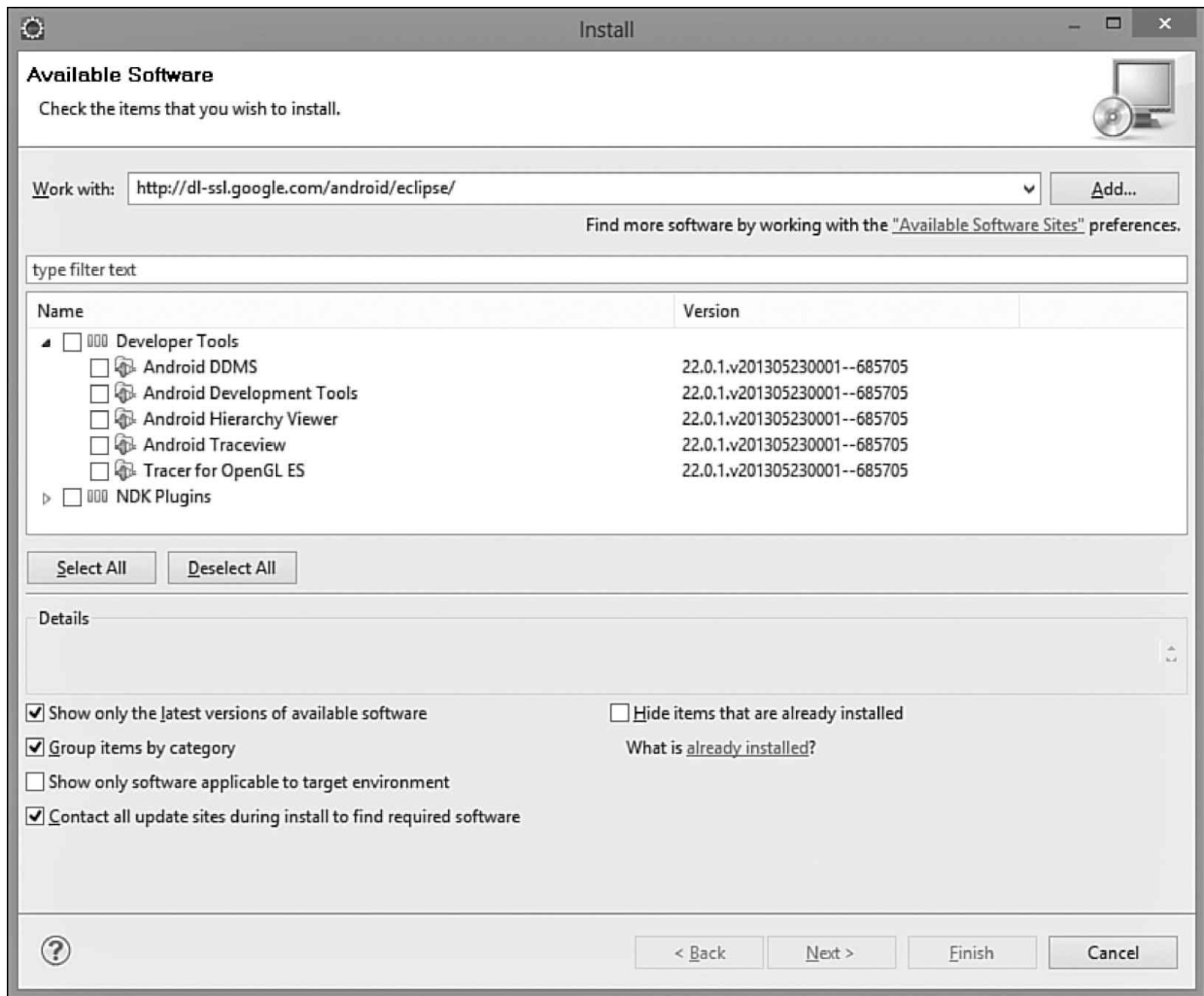
第一次运行Eclipse时，它必须设置一个插件，以将IDE与Android SDK集成。这个插件在IED界面中添加了一个与Android相关的菜单命令，这样也就可以创建和管理Android app了。要实现该目的，必须在计算机上以管理员的权限运行Eclipse。在Window 7或Windows 8中，默认是以非管理员的身份来运行程序，因此当你单击Eclipse时，它无法运行。相反，要右键单击其快捷方式，然后选择Run As Administrator。

Eclipse会自动设置Android，并显示一个欢迎界面。该界面中带有New Android Application按钮，以及几个链接，通过这几个链接可以获悉Android编程的更多知识。

对Eclipse和Android SDK来说，你都应该定期检查其更新。Android在以迅猛的速度发展，新的手机和其他设备也涌入市场，因此SDK必须能够支持这些新设备。

选择菜单命令Help->Check for Updates，查看Eclipse是否有更新。

选择Window->SDK Manager，运行Android SDK管理器，这也会查找SDK更新。在你需要更新的包上，管理器会显示一个复选框，如图D.1所示。



图D.1 为Android SDK安装新的包

有些包会被自动选中，这表示管理器建议将其更新。按照它的建议来即可。当选择了要更新的包，剔除了不想更新的包后，单击Install Packages。

在安装完SDK更新后，系统将要求关闭SDK管理器，然后再检查Eclipse更新。

Watch Out!

警告：

如果更新过程因为遇到错误而失败（尤其是下载文件时遇到的“access denied”问题），关闭Eclipse然后以管理员权限再次运行。这通常可以解决问题。

D.4 设置你的手机

Android SDK包含一个模拟器，它的行为与Android手机相似，可以运行你创建的app。当你编写app时，这会带来极大的方便，因为尽管你可以让你的app在运行在测试环境中，但是某些时候，你还需要查看一下它是如何在真实的Android手机（或其他设备）上运行的。

通过计算机的USB连接，可以将使用SDK编写的app部署在Android设备中。USB连接也可以用来向设备传输文件，或者将设备中的文件传输到计算机。

在连接USB线缆之前，必须执行如下步骤，以在手机上启用USB调试。

1. 在手机上运行Setting app，这通常是位于手机主界面上的一个app；也可以在完整的应用列表中找到它。

2. 运行Setting app，选择Applications（如果有必要），然后查找Developmnet或DeveloperOptions按钮。将其选中，然后选择USB调试盒。

By the Way

注意

在Android 4.2或更高的版本中，Developer Options可能会隐藏起来。要使其可见，选择Setting->About Phone，然后敲击Build Number 多次即可。

在其他设备中，该选项可能位于设置中的其他位置，其名字可能为USB连接模式、USB调试或其他名字。Android站点<http://developer.android.com> 中有一个文档，它讲解了如何为不同的Android设备设置该选项。

将USB线缆的一端连接到计算机，另一端连接到你的手机。则在设备的顶部会出现一个像bug一样的小Android图标。该图标与显示时间、连接图标和电池图标相邻。

向下拖动顶部栏，将看到USB Debugging Connected和USB Connected消息（见图D.2）。

07/26/2011



12:03 AM



Wi-Fi



Bluetooth



GPS



Silent



4G

Sprint

Ongoing



USB debugging connected

Select to disable USB debugging.



USB connected

Select to copy files to/from your computer.

Notifications



New voicemail

122 unheard voicemails

7:53 PM



图D.2 在USB调试模式中使用Android手机

此时，手机设置完毕，但是你的计算机也需要一定的配置，才能连接到该设备。如果之前从来没有通过USB线缆将手机连接到计算机上，则查看你的手机说明书，以获悉其操作方法。你可能需要从随手机销售的CD中安装驱动程序，或者是从手机生产商的网站上下载该驱动程序。

在Windows中，Android SDK在Eclipse内运行，你可以使用它来下载USB驱动程序包，这是一个驱动程序集，可以用于不同的手机和其他设备，同时还可以下载其他与设备相关的包。选择Window->Android SDK，查看哪些包可以使用。

在第24章，我们使用Android开发工具来创建和运行Android app。如果所有的设置没有问题，则它可以成功地运行在模拟器和Android手机上。

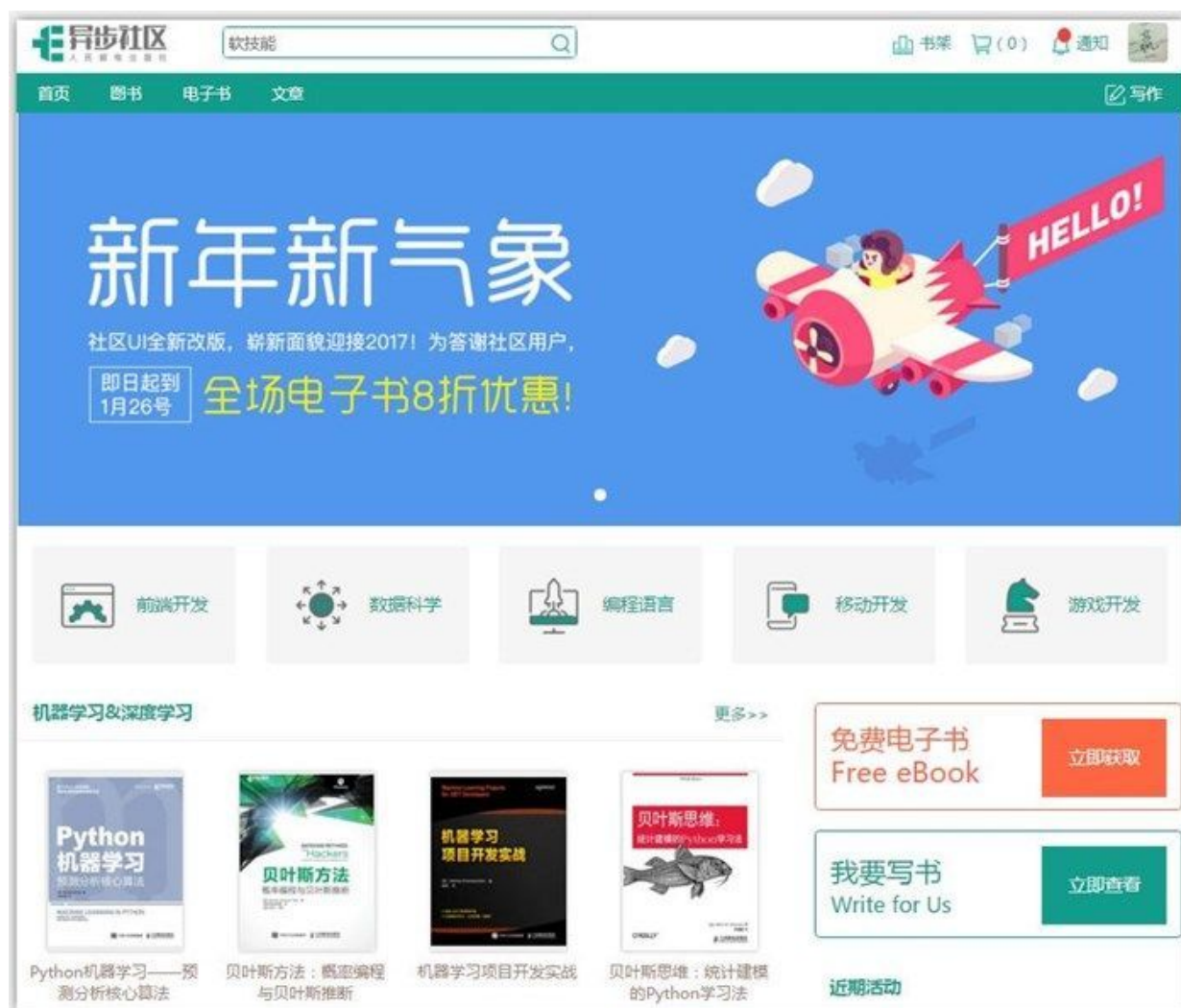
Android的官方文档提供了更多的指导，来帮助你设置手机，其地址为[http://developer. android.com/training/basics/firstapp](http://developer.android.com/training/basics/firstapp)。

欢迎来到异步社区！

异步社区的来历

异步社区(www.epubit.com.cn)是人民邮电出版社旗下IT专业图书旗舰社区，于2015年8月上线运营。

异步社区依托于人民邮电出版社20余年的IT专业优质出版资源和编辑策划团队，打造传统出版与电子出版和自出版结合、纸质书与电子书结合、传统印刷与POD按需印刷结合的出版平台，提供最新技术资讯，为作者和读者打造交流互动的平台。



社区里都有什么？

购买图书

我们出版的图书涵盖主流IT技术，在编程语言、Web技术、数据科学等领域有众多经典畅销图书。社区现已上线图书1000余种，电子书400多种，部分新书实现纸书、电子书同步出版。我们还会定期发布新书书讯。

下载资源

社区内提供随书附赠的资源，如书中的案例或程序源代码。

另外，社区还提供了大量的免费电子书，只要注册成为社区用户就可以免费下载。

与作译者互动

很多图书的作译者已经入驻社区，您可以关注他们，咨询技术问题；可以阅读不断更新的技术文章，听作译者和编辑畅聊好书背后有趣的故事；还可以参与社区的作者访谈栏目，向您关注的作者提出采访题目。

灵活优惠的购书

您可以方便地下单购买纸质图书或电子图书，纸质图书直接从人民邮电出版社书库发货，电子书提供多种阅读格式。

对于重磅新书，社区提供预售和新书首发服务，用户可以第一时间买到心仪的新书。

用户帐户中的积分可以用于购书优惠。100积分=1元，购买图书时，在 里填入可使用的积分数值，即可扣减相应金额。

特别优惠

购买本电子书的读者专享**异步社区优惠券**。使用方法：注册成为社区用户，在下单购书时输入“57AWG”，然后点击“使用优惠码”，即可享受电子书8折优惠（本优惠券只可使用一次）。

纸电图书组合购买

社区独家提供纸质图书和电子书组合购买方式，价格优惠，一次购买，多种阅读选择。

The screenshot displays the book page for "Wireshark网络分析的艺术" (The Art of Network Analysis with Wireshark). The page includes the book cover, author information (林沛满), and a detailed description. It offers three purchase options: a discounted paper edition (¥31.50), an electronic edition (¥25.00), and a combined bundle (¥45.00). A bundle of two paper editions is also shown for a total price of 75.60. The page also features a sidebar with the author's profile (LinPeiman), a list of reviews, and a section for recommended books like "Nmap渗透测试指南".

Wireshark网络分析的艺术

作者：林沛满
责编：傅道坤
分类：计算机科学 > 安全与加密 > 网络安全

Wireshark是当前最流行的网络包分析工具。它上手简单，无需培训就可入门。很多棘手的网络问题遇到Wireshark都能迎刃而解。本书挑选的网络包来自真实场景，经典且接地气。讲解时采用了生活化的

5.6K 浏览 57 想法 7 推荐

下载PDF样章 配套文件下载

分享：

纸质 ¥45.00-¥31.50 (7折) 电子 ¥25.00 电子+纸质 ¥45.00

购买

纸质 (纸质) + 纸质 (纸质) 总价：75.60 一起购买

目录 评论 9 勘误 1 出版信息

作者简介 专业书评 内容提要

本书作者：LinPeiman 上海 1.0K经验值

发私信 送积分 关注

《Wireshark网络分析就这么简单》即《Wireshark网络分析的艺术》作者

兑换样书 立即兑换 如何赚取积分

电子书版本 PDF Epub Mobi

精彩推荐 Nmap渗透测试指南 作者：南广明

社区里还可以做什么？

提交勘误

您可以在图书页面下方提交勘误，每条勘误被确认后可以获得100积分。热心勘误的读者还有机会参与书稿的审校和翻译工作。

写作

社区提供基于Markdown的写作环境，喜欢写作的您可以在此一试身手，在社区里分享您的技术心得和读书体会，更可以体验自出版的乐趣，轻松实现出版梦想。

如果成为社区认证作译者，还可以享受异步社区提供的作者专享特色服务。

会议活动早知道

您可以掌握IT圈的技术会议资讯，更有机会免费获赠大会门票。

加入异步

扫描任意二维码都能找到我们：



异步社区



微信订阅号



微信服务号



官方微博



QQ群: 436746675

社区网址: www.epubit.com.cn

官方微信: 异步社区

官方微博： @人邮异步社区， @人民邮电出版社-信息技术分社

投稿&咨询： contact@epubit.com.cn